ENGINEERING DESIGN OPTIMIZATION BY THE AUGMENTED LAGRANGE MULTIPLIER METHOD

Leroy E. Madsen

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ENGINEERING DESIGN OPTIMIZATION
BY THE AUGMENTED LAGRANGE MULTIPLIER METHOD

by

Leroy E. Madsen

March 1981

Thesis Advisor: G. N. Vanderplaats

Approved for public release; distribution unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Engineering Design Optimization by the Augmented Lagrange Multiplier Method		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1981 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Leroy E. Madsen		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1981 13. NUMBER OF PAGES 89 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Augmented Lagrange Multiplier Method, engineering design optimization, equality and inequality constraints. structural design optimization, linear convergence.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer program was developed for solving equality and inequality constrained optimization problems by the Augmented Lagrange Multiplier method. The program was developed specifically for use in engineering design.

The historical evolution and theoretical development of the multiplier method is presented. Several examples are used to demonstrate the effects of penalty parameters and multipliers on

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Item 20 (contd)

he convergence and accuracy of the method. Computational experi-
nce with variations to the method is documented.

A brief literature search of the multiplier method's application
o engineering design is summarized. The method is demonstrated
ith several mathematical and engineering examples. A comparison
o classical penalty methods and the method of feasible directions
as performed in each case.

Approved for public release; distribution unlimited

Engineering Design Optimization by the Augmented Lagrange Multiplier Method

by

Leroy E. Madsen
Lieutenant, United States Navy
B.S.E.E., Purdue University, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1981

ABSTRACT

A computer program was developed for solving equality and inequality constrained optimization problems by the Augmented Lagrange Multiplier method. The program was developed specifically for use in engineering design.

The historical evolution and theoretical development of the multiplier method is presented. Several examples are used to demonstrate the effects of penalty parameters and multipliers on the convergence and accuracy of the method. Computational experience with variations to the method is documented.

A brief literature search of the multiplier method's application to engineering design is summarized. The method is demonstrated with several mathematical and engineering examples. A comparison to classical penalty methods and the method of feasible directions was performed in each case.

TABLE OF CONTENTS

I. INTRODUCTION

Several methods currently exist which solve the function minimization problem. While each has its advantage in specialized cases, no one method, including the multiplier method, is universally more efficient and accurate than the others. The Augmented Lagrange Multiplier method has been shown to be an efficient and accurate method of engineering design optimization. The multiplier method is particularly advantageous in nonlinear equality constrained problems. It has become increasingly popular in engineering design optimization because of its good convergence rate and its attractive theoretical properties.

The general nonlinear problem is defined:

$$\text{Minimize } f(\bar{X}) \tag{1.1a}$$

$$\text{subject to } g_i(\bar{X}) \leq 0 \quad (i=1,\ldots,\ell) \tag{1.1b}$$

$$\text{and } h_j(\bar{X}) = 0 \quad (j=1,\ldots,m<n) \tag{1.1c}$$

$$\text{and } X_i^{\ell} < X_i < X_i^{u} \quad (i=1,\ldots,n) \tag{1.1d}$$

where $\bar{X}$ is a vector of n design variables, $f(\bar{X})$ is the objective function, $g(\bar{X})$ is a set of ℓ inequality constraint functions, and $h(\bar{X})$ is a set of m equality constraint functions, and X_i^{ℓ} and X_i^{u} are bounds on the design variables, referred to as side constraints. The problem is solved by creating a single augmented Lagrange function $L(\bar{X},\bar{\lambda})$, where $\bar{\lambda}$ is a vector of n Lagrange multipliers. The optimal solution $(\bar{X}^*,\bar{\lambda}^*)$ is

found by alternately solving a series of unconstrained minimizations followed by a simple update of the multipliers, $\bar{\lambda}$.

In this study, the unconstrained minimization is performed by the Davidon-Fletcher-Powell method [1] utilizing a combination of the Golden Section and polynomial one-dimensional search procedures. The Davidon-Fletcher-Powell method provides good reliability and convergence properties. The Golden Section search provides a reliable means of reducing the search bounds. In most cases, polynomial interpolation obtains a more accurate minimum than Golden Section for sufficiently narrow bounds. The combination provided a reliable and accurate one-dimensional search.

The multiplier method has significant advantages. As will be shown later, the inherent numerical ill-conditioning of more common penalty function methods is reduced in the multiplier method. Exact solutions, not possible in penalty methods, are also attainable. Finally, multiplier methods are not restricted to convex programming as are pure primal-dual methods.

There have been numerous applications of the multiplier method in the various disciplines of engineering design optimization. Imai [2] effectively used the method in structural optimization. Fax and Mills [3], while limited to equality constraints, applied the method to heat exchanger optimization. Hedderich's [4] work in heat exchanger

optimization showed the need for a method which could treat equality constraints directly, in addition to inequality constraints. He showed the Constrained function Minimization Program, CONMIN [5] to be very inefficient for the equality constrained problem. It is his work that motivated this research of the multiplier method.

It is the objective of this research to develop an operational computer program applicable to the various disciplines of engineering design optimization. It is not the author's intent to develop a universally superior optimization program, but to develop a program to be incorporated into a library of optimization programs with flexibility in the choice of unconstrained minimization subprograms, one dimensional search subprograms, derivative evaluation subprograms, and convergence criterion.

The theory and historical development of the multiplier method is presented, concluded by a multiplier method algorithm for nonlinear equality and inequality constrained problems. Experimentation with computational aspects is then summarized. Finally, a set of mathematical and engineering test cases are solved demonstrating the method's effectiveness, followed by conclusions and a discussion of results.

II. THE MULTIPLIER METHOD

The development of the multiplier method and a brief mathematical background is presented in this chapter. Mathematical proofs are brief since computational aspects and application of the method are more the concern of this thesis. This chapter includes historical background of the method, a review of the Lagrange multiplier and penalty function methods, and a development of the algorithm for equality and inequality constrained problems. Rigorous mathematical development of the method is given by Hestenes [6].

A. BACKGROUND

The multiplier method was independently introduced in 1968 by Hestenes [7] and Powell [8]. Both developed an augmented Lagrange function and solved a series of unconstrained minimizations followed by a simple update of the multiplier vector. Powell showed that, if the function had continuous second derivatives, the method would converge locally at a linear rate while the penalty parameter remained finite. As will be shown later, this eliminated the inherent numerical ill-conditioning of classical penalty function methods.

Extensive testing and modification has been done to the method of Hestenes and Powell. Miele et. al. [9,10] applied the method extensively to equality constrained problems. Tripathi and Navandra [11] experimented with updating the multiplier after each one dimensional search instead of after each unconstrained minimization. Rockafellar [12,13] expanded the method to nonlinear inequality constrained problems. Global convergence of the method was proved by Rockafellar [13] for convex programming problems. Bertsekas [14,15] provides convergence proofs for the general nonlinear problem.

B. THE LAGRANGIAN

The general nonlinear equality constrained problem is defined:

$$\text{Minimize } f(\bar{X}) \tag{2.1a}$$

$$\text{subject to } h_i(\bar{X}) = 0 \quad (i=1,\ldots,m<n) \tag{2.1b}$$

The theorem associated with Lagrange multiplier method states that:

"If $\bar{X}^*$ affords a local minimum to $f(\bar{X})$ subject to the constraints $h_i(\bar{X})=0$, then there exists a unique set of multipliers, λ_i, $(i=1,\ldots,m)$ such that if

$$L(\bar{X},\bar{\lambda}) = f(\bar{X}) + \sum_{i=1}^{m} \lambda_i h_i(\bar{X}), \tag{2.2a}$$

then

$$\nabla L(\bar{X}^*,\bar{\lambda}^*) = \nabla f(\bar{X}^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\bar{X}^*) = 0 \tag{2.2b}$$

and

$$L''(\bar{X}^*,\bar{\lambda}^*) = \frac{\partial^2 L(X_i^*,\lambda_i^*)}{\partial^2 X_i} > 0 \tag{2.2c}$$

where ∇ denotes the gradient of the function and L" denotes the second derivative of L"[6].

Equations 2.2b and 2.2c are the necessary conditions for locally constrained minima. Equation 2.2b and the feasibility condition (Eq. 2.1b) constitute the Kuhn-Tucker necessary conditions for optimality [6]. It is assumed that $f(\bar{X})$ and $h_i(\bar{X})$ are second order differentiable and that the gradients $\nabla h_i(\bar{X})$ are not zero at $\bar{X}^*$.

The problem can now be stated in terms of the equivalent classical Lagrangian.

$$\text{Minimize } L(\bar{X},\bar{\lambda}) \tag{2.3a}$$

$$\text{subject to } h_i(\bar{X}) = 0 \quad (i=1,\ldots,m<n) \tag{2.3b}$$

Assuming the existence of the saddle points of the Lagrangian $L(\bar{X},\bar{\lambda})$, the condition exists:

$$L(\bar{X}^*,\bar{\lambda}) \leq L(\bar{X}^*,\bar{\lambda}^*) \leq L(\bar{X}, \bar{\lambda}^*) \tag{2.4}$$

The optimal pair $(\bar{X}^*,\bar{\lambda}^*)$ can be obtained by first minimizing $L(\bar{X},\bar{\lambda})$ respect to $\bar{X}$, then maximizing $L(\bar{X}^k,\bar{\lambda})$ with respect to λ by update equation,

$$\lambda_i^{k+1} = \lambda_i^k + c[h_i(\bar{X}^k)] \tag{2.5}$$

where c is a scalar parameter (stepsize), k is the iteration number, and $\bar{X}^k$ is the local minimum of $L(\bar{X}^k,\bar{\lambda}^k)$. The procedure is repeated until convergence is attained. This is the so called "primal-dual" method.

Serious disadvantages are encountered in the primal-dual method. First, the problem (Eq. 2.3) must have a locally convex structure for the dual problem to be well defined and

for Equation 2.5 to be meaningful [16]. Second, a large number of iterations are usually required to minimize $L(\bar{X},\bar{\lambda})$ (Eq. 2.2a) since the ascent iteration (Eq. 2.5) converges only moderately fast. Thus, primal-dual methods have found application in only a limited class of problems where minimization of the Lagrangian (Eq. 2.2a) can be efficiently carried out due to special structure, as shown by Luenberger [16], or where the design problem exhibits a unique form, as shown by Schmit and Fleury [17].

C. PENALTY FUNCTION METHODS

Penalty function methods have been used extensively since the mid-1940's [18]. They are considered to be efficient for inequality constrained problems.

Given the nonlinear inequality constrained problem:

$$\text{Minimize } f(\bar{X}) \tag{2.6a}$$

$$\text{subject to } g_i(\bar{X}) \leq 0 \quad (i=1,\ldots,\ell) \tag{2.6b}$$

The general exterior penalty function is defined:

$$F_e(\bar{X},c) = f(\bar{X}) + c\sum_{i=1}^{\ell} \phi(t) \tag{2.7}$$

where $\phi(t)$ is some scalar penalty function of the constraints and c is some scalar penalty parameter. The most common penalty function is the quadratic $\phi(t)=t^2/2$. However, it may be desirable at times to use other penalty functions. In this study the quadratic penalty function is used such that Equation 2.7 becomes

$$F_e(\bar{X},c) = f(\bar{X}) + \frac{c}{2} \sum_{i=1}^{\ell} p_i^2(\bar{X}) \tag{2.8}$$

where $P_i(\bar{X}) = \begin{cases} g_i(\bar{X}), & \text{if } g_i(\bar{X}) \geq 0 \\ 0 \quad , & \text{otherwise} \end{cases}$

The problem is now an unconstrained minimization of $F_e(\bar{X})$. The convergence of the method is easily seen by a simple numerical example.

Example 1 [2]. Minimize x such that $1-x \leq 0$.

Solution: First, examine the solution to the problem by primal-dual method. Equation 2.2a becomes

$$L(x,\lambda) = x + \lambda(1-x) \tag{2.9}$$

where λ is a real non-negative number. From the stationary conditions of Equation 2.9:

$$(x^*,\lambda^*) = (1,1)$$

Solving by exterior penalty method, Equation 2.8 becomes

$$F_e(x,c) = x + \frac{c}{2} p^2(x) \tag{2.10}$$

where $p(x) = \begin{cases} 1 - x, & x \geq 1 \\ 0 \quad , & \text{otherwise} \end{cases}$

From the stationary conditions, a minimum exists at

$$x' = 1 - \frac{1}{c}$$

To obtain the optimal x*, a series of unconstrained minimizations are solved while increasing the penalty parameter c toward infinity. It is apparent that as $c \to \infty$, $x' \to x^*$. No exact solution is obtained and the optimum is approached from the infeasible region.

The plots of $F_e(x)$ vs. x and $f(x)$ vs. c are given in Figs. 1 and 2, respectively. Note in Fig. 1, as c gets larger, the function changes more rapidly and the optimum becomes more difficult to find regardless of the minimization technique. This is a cause of numerical ill-conditioning inherent with penalty methods. Figure 2 shows the asymtotic convergence of the method.

Interior penalty functions have the advantage of approaching the optimum from the feasible region thus yielding a feasible solution. However, the penalty function is discontinuous at the constraint boundaries. Also, the same problems of ill-conditioning and slow convergence exist as seen by Example 2.

Example 2. Solve Example 1 by the interior penalty function method.

Solution: One form of the interior penalty function is

$$F_{int}(x) = f(x) - \frac{1}{c} \sum_{i=1}^{m} \frac{1}{g_i(x)} \tag{2.11}$$

Substituting,

$$F_{int}(x) = -\frac{1}{c} \frac{1}{1-x} \tag{2.12}$$

It can be shown analytically that a minimum exists at

$$x' = 1 + \frac{1}{c}$$

It is again apparent that $x' \to x^*$ only as $c \to \infty$.

Figures 3 and 4 illustrate that the same problems of ill-conditioning and slow convergence exist as with the exterior

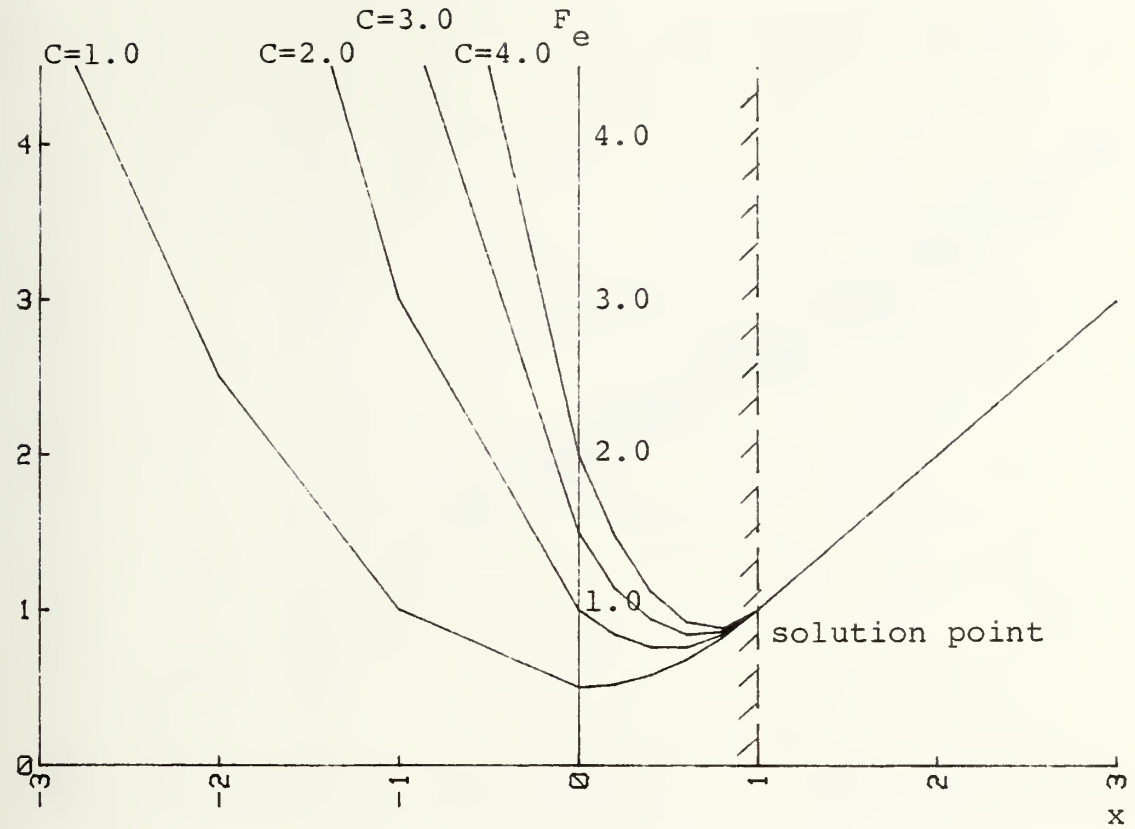


Fig. 1. Exterior Quadratic Penalty Function

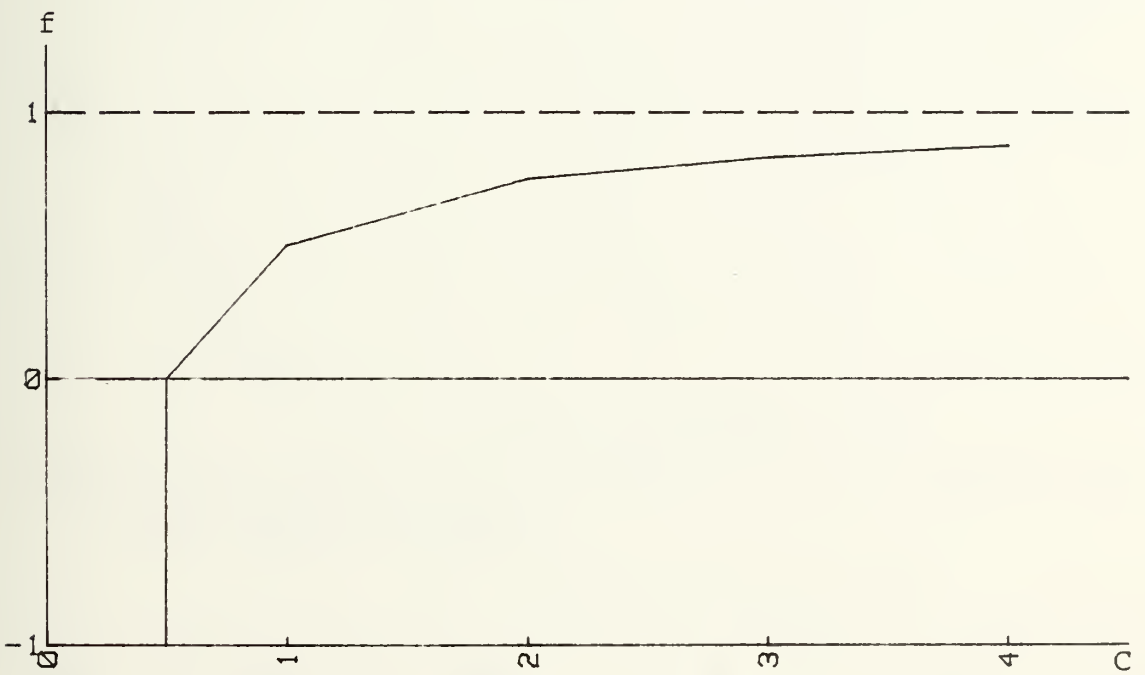


Fig. 2. Convergence of Exterior Penalty Function

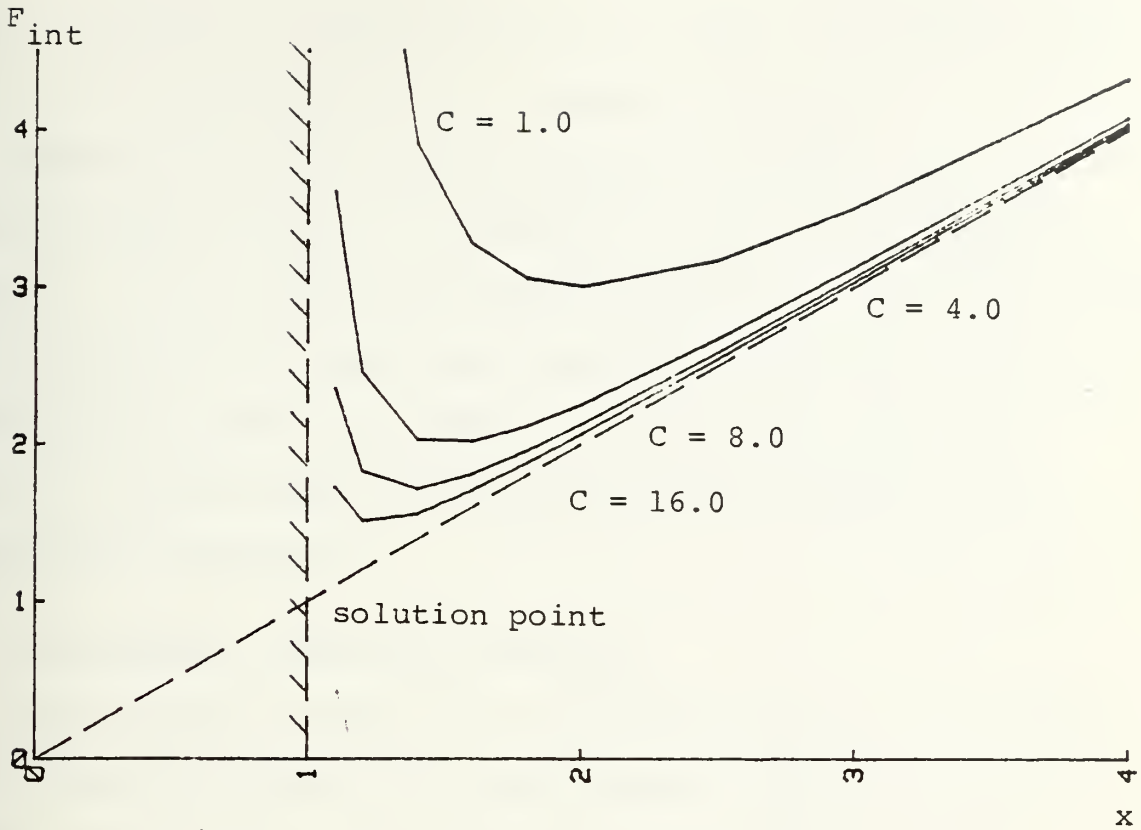


Fig. 3. Interior Penalty Function

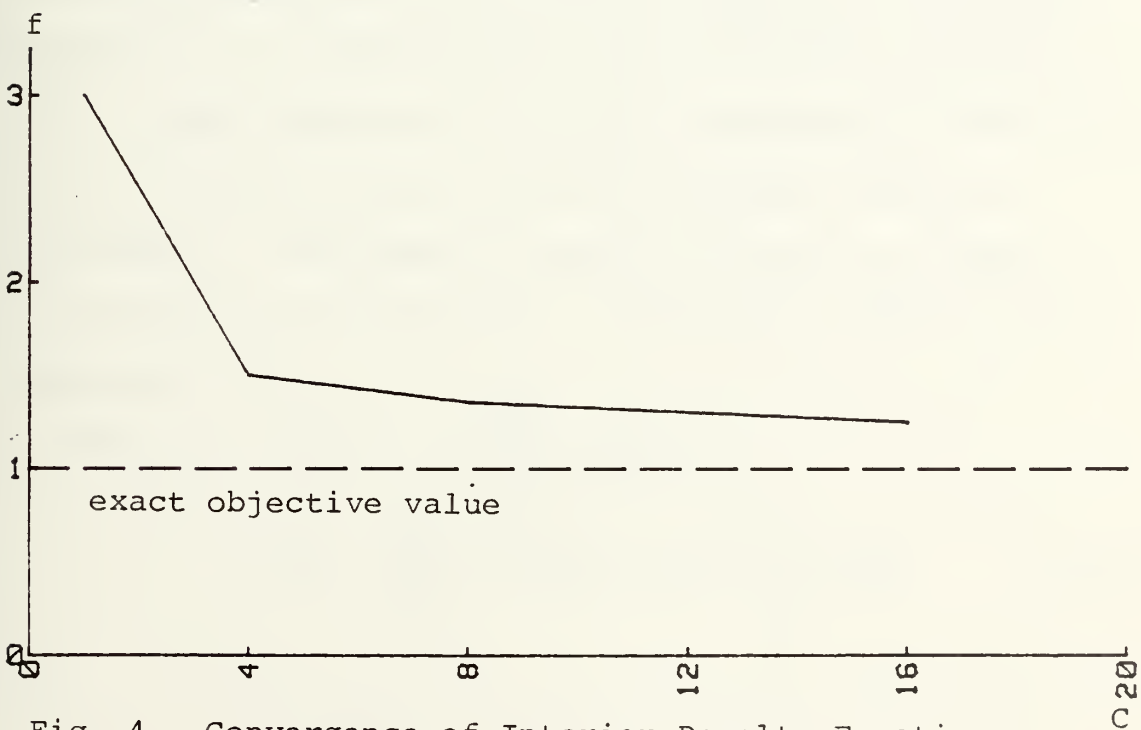


Fig. 4. Convergence of Interior Penalty Function

penalty method. An exact solution is again not possible but the solution achieved here is, however, feasible.

Extended interior penalty methods [19] avoid function discontinuity at $g_i(\bar{X}) = 0$ inherent in the interior penalty method. It is, therefore, less susceptible to ill-conditioning. It does exhibit the same slow convergence of other penalty methods due to the requirement to increase c to infinity. It is for these reasons that the multiplier method is an attractive alternative.

D. THE EQUALITY CONSTRAINED PROBLEM

The multiplier method can be perceived to be a combined primal-dual and penalty function methods. Though they are theoretically similar, their behavior is quite different.

It has been shown that the original equality constrained problem (Eq. 2.1) is equivalent to the classical Lagrangian (Eq. 2.3). Since Equation 2.3 is still an equality constrained problem, it can be solved by the usual exterior penalty function method. The quadratic penalty function is used so first derivatives are continuous. Substituting Equation 2.3 into Equation 2.8:

$$A(\bar{X},\bar{\lambda},c) = L(\bar{X},\bar{\lambda}) + \frac{c}{2} \sum_{i=1}^{m} p_i^2(\bar{x})$$

$$= f(\bar{X}) + \sum_{i=1}^{m} \lambda_i h_i(\bar{X}) + \frac{c}{2} \sum_{i=1}^{m} p_i^2(\bar{X}) \qquad (2.13)$$

where $p_i(\bar{X}) = h_i(\bar{X})$

Equation 2.13 is defined to be the "Augmented Lagrange function" for the equality constrained problem.

By the nature of penalty function methods, as c goes to infinity, Equation 2.13 converges to the solution of Equation 2.8. Concurrently, by choosing an appropriate value of $\bar{\lambda}$ with a suitable update formula, Equation 2.13 can be solved by a series of unconstrained minimizations to obtain a solution to the original problem (Eq. 2.1).

The selection of $\bar{\lambda}$ can be significant to the behavior of the function as seen in the following two extreme cases. First, take $\lambda_i=0$ for all unconstrained minimizations. Equation 2.13 becomes

$$A(\bar{X},\bar{\lambda},c) = f(\bar{X}) + \frac{c}{2} \sum_{i=1}^{m} h_i^2(\bar{x}) \qquad (2.14)$$

which is the usual quadratic exterior penalty function (Eq. 2.8). It has previously been shown that the function only converges to a minimum $f(\bar{X}^*)$ as c goes to infinity. It has also been shown to be slow in converging, susceptible to numerical ill-conditioning, and to attain only a near optimum solution from the infeasible region.

Next, consider the case where $\lambda_i^o=\lambda_i^*$. At the minimum, the stationary condition requires that

$$\nabla A(\bar{X}^*,\bar{\lambda}^*,c) = \nabla f(\bar{X}^*) + \sum_{i=1}^{m} \lambda_i h_i(\bar{X}^*)$$

$$= \varepsilon \sum_{i=1}^{m} h_i(\bar{X}^*) \nabla h(\bar{X}^*) = 0 \qquad (2.15)$$

The feasibility condition $h_i(\bar{X}^*)=0$ implies that Equation 2.13 is independent of the value of c. This leads to two important results. First, if the optimum $\bar{\lambda}^*$ is known initially, the solution can be obtained in one unconstrained minimization. Second, since $A(\bar{X},\bar{\lambda},c)$ is independent of c at the optimum, it is not necessary to sequentially increase c to infinity to attain a solution. The second result implies that a finite c can be chosen, thus avoiding the inherent ill-conditioning of the penalty methods.

The task is now to find initial values for c and $\bar{\lambda}$ with sequential update formulas for each to achieve a suitable rate of convergence. Collecting terms, Equation 2.15 becomes

$$\nabla f(\bar{X}^*) + \sum_{i=1}^{m} [\lambda_i^* + ch_i(\bar{X}^*)] \nabla h_i(\bar{X}^*) = 0 \qquad (2.16)$$

By the Kuhn-Tucker conditions, at the optimum $(\bar{X}^*,\bar{\lambda}^*)$.

$$\nabla f(\bar{X}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(\bar{X}^*) = 0 \qquad (2.17)$$

Thus, Equation 2.16 reduces to Equation 2.17 in the limit. This implies as update formula for $\bar{\lambda}$ such that

$$\lambda_i^{k+1} = \lambda_i^k + ch_i(\bar{X}^k) \qquad (2.18)$$

where $\bar{X}^k$ is the solution to the k^{th} unconstrained minimization. Proposed initially by Hestenes [6], Equation 2.18 remains the most popular update formula for $\bar{\lambda}$.

Little experimentation has been done with choosing and initial $\bar{\lambda}^o$. As shown earlier, if $\bar{\lambda}^o=\bar{\lambda}^*$ the solution is obtained in one unconstrained minimization. Obviously, the closer $\bar{\lambda}^o$ is to $\bar{\lambda}^*$, the more rapid the convergence. A widely accepted practice is to choose $\lambda_i^o=0$ due to the computational convenience, and because no other multiplier has consistently proven more efficient. Active and violated constraints are immediately identified in this case since a non-zero λ_i can only occur for an active or violated constraint. A constraint is active if the multiplier is at its optimum $\lambda_i^*\neq 0$ when $\bar{X}\rightarrow\bar{X}^*$. This eliminates any extra computations to check constraint behavior.

Before determining the choice of the initial penalty parameter c^o and the update formula for c, it is first convenient to examine the convergence of the method which is directly related to the choice of c. Various proofs of linear convergence to a local minimum have been developed [8,20,21]. Rockafellar [13] proved global convergence of the method for convex programming. Bertsekas [14,15] provides a rigorous proof of the method's convergence for the general nonlinear problem. He compares the convergence of the various multiplier methods, penalty methods, and primal-dual methods [22]. His results for the Hestenes' [6] multiplier method are summarized here.

Recall the assumptions made for a local minimum to exist:

Assumption 1. There exists a local minimizing point $\bar{X}^k$ of problem 2.1 which satisfies the second

order sufficiency conditions for an isolated local minimum, i.e., f and h_i are twice continuously differentiable in a neighborhood of $\bar{X}^k$, the gradients $\Delta h_i(\bar{X}^k)$, (i=1,...,m) are linearly independent and there exists a Lagrange multiplier vector $\bar{\lambda}$, such that $\nabla L(\bar{X}^k,\bar{\lambda}^k)=0$ and $|\nabla^2 L(\bar{X}^k,\bar{\lambda}^k)|>0$ [22].

Assumption 2. The penalty function $\phi(t)$ is twice continuously differentable in an open interval containing zero and $\phi''(0)>0$, where ϕ'' denotes the second derivative [22].

Also, assuming the Hessian matrices $\nabla^2 f(\bar{X})$, $\nabla^2 h_i(\bar{X})$, and the second derivative ϕ'' are Lipschitz continuous,[1] there exists a scalar $c^*>0$, and $M>0$ such that for every $c^k>c^*$ the function $L_c(\bar{X},\bar{\lambda})$ has a unique minimum $\bar{X}^k(\bar{\lambda},c)$. Furthermore,

$$||\bar{X}^k - \bar{X}^*|| \leq \frac{M||\bar{\lambda}^k-\bar{\lambda}^*||}{c} \tag{2.19}$$

and

$$||\bar{\lambda}^{k+1} - \bar{\lambda}^*|| \leq \frac{M||\bar{\lambda}^k-\lambda^*||}{c} \tag{2.20}$$

where $\lambda_i^{k+1} = \lambda_i^k + c\, h_i(\bar{X}^k)$

The notation $||\cdot||$ denotes the usual Euclidean norm.

From Equation 2.20, if $c^k \to c < \infty$ for the non-decreasing positive penalty sequence $\{c^k\}$ so as to ensure $M/c^k \to 1$, then the sequence $\{\bar{\lambda}^k\}$ converges to λ^* linearly. If c^k goes to infinity,

---

[1] A function is said to satisfy a Lipschitz condition of order m on the closed interval (a,b) if there exists a constant c such that $|f(x_2)-f(x_1)|<c|x_2-x_1|$ for all values x_1, x_2 on (a,b).

$M/c^k \to 0$, and the sequence $\{\bar{\lambda}^k\}$ converges superlinearly. From Equation 2.19, for sufficiently large but finite c^k, the sequence $\{\bar{X}^k\}$ converges to X* since $\{\bar{X}^k\}$ converges. If c^k goes to infinity, $\{\bar{X}^k\}$ converges by the same argument.

The rate of convergence of $\{\bar{X}^k\}$ is represented by the right hand side of Equation 2.19. Since $\{\bar{\lambda}^k\}$ converges at least linearly, $\{\bar{X}^k\}$ converges at least linearly. This has been shown to be significantly faster than pure penalty or primal-dual methods. Numerical ill-conditioning is avoided since this linear convergence rate is achieved with a sufficiently large finite c.

The convergence criterion of Equations 2.19 and 2.20 are global in nature since no bounds restriction has been imposed. No restrictions on the convex or non-convex nature of the problem are specified as well. The global convergence of the multiplier method is contingent upon the ability of the unconstrained minimization method to generate a sequence $\{\bar{X}^k\}$ which are well defined local minimums to the function $A(\bar{X}^k, \bar{\lambda}^k, c)$. Naturally, the function $A(\bar{X}^k, \bar{\lambda}^k, c)$ may have other local minimums to which the unconstrained minimization method may be attracted. Unless the unconstrained minimization method stays in the neighborhood of the same local minimum, the convergence argument is invalid and there is no assurance that the multiplier method will perform any better or worse than the penalty methods. It should be noted that the usual practice is to use the last point $\bar{X}^k$ of the k^{th} minimization

as the starting point of the (k+1)th minimization. This generally produces sequences $\{\bar{x}^k\}$ which are close to the same local minimum of $A(\bar{X},\bar{\lambda},c)$.

The convergence rate of the multiplier iteration (Eq. 2.18) is linear with the convergence ratio essentially inversely proportional to the penalty parameter c [22]. This fact is strongly dependent on Assumptions 1 and 2. If either of the assumptions is relaxed, the convergence rate may become sublinear or superlinear as the following examples show.

Example 3 [22]. Consider the scalar problem minimize $x^2/2$ such that $x=0$ with an optimal solution $x^*=0$, $\lambda^*=0$. In this example Assumption 2 is not satisfied. For $\lambda \leq 0$, $\phi(t)=|t|^3/3$, and c=1, Equation 2.13 becomes

$$x^1(\lambda,1) = \frac{-1-\sqrt{(1-4\lambda)}}{2}$$

Starting at $\lambda^o=0$, Equation 2.18 becomes

$$\lambda^{k+1} = \frac{1-\sqrt{(1-4\lambda^k)}}{2}$$

The $\lim_{k\to\infty}(\frac{\lambda^{k+1}}{\lambda^k}) = 1$, i.e., a sublinear convergence rate.

Example 4 [22]. Consider Example 3 where $\phi(t)=2\sqrt{|t|}/3$, c=1. Again, Assumption 2 is not satisfied. The solution to Equation 2.13 becomes

$$x^1(\lambda,1) = \frac{(-1+\sqrt{(1-4\lambda)})^2}{4}$$

Starting at $\lambda^0=0$, Equation 2.18 becomes

$$\lambda^{k+1} = \lambda^k + \frac{(-1 + \sqrt{1-4\lambda^k})}{2}$$

It can be shown that

$$\lim_{k\to\infty} \frac{|\lambda^{k+1}|}{(\lambda^k)^2} = 1$$

hence, a superlinear convergence (order 2).

Example 5 [22]. Consider the problem, minimize $|x|^3/2$ such that $x=0$. Again $x^*=\lambda^*=0$, but Assumption 1 is not satisfied. For $\phi(t)=t^2/2$, $c=1$, the solution to Equation 2.13 becomes

$$x^k(\lambda,1) = \frac{(-1+\sqrt{1-4\lambda})}{2}$$

and Equation 2.18 becomes

$$\lambda^{k+1} = \lambda^k + \frac{(-1 + \sqrt{1-4\lambda^k})}{2}$$

Again,

$$\lim_{k\to\infty} \frac{|\lambda^{k-1}|}{(\lambda^k)^2} = 1$$

hence, superlinear convergence.

It can be shown that the convergence rate is influenced substantially by the rates of change of the derivatives (curvature) of the primal function $p(u) = \min f(\bar{X})$, where $u = h(\bar{X})$, and the penalized primal function derivatives $p(u)+c[\phi(u)]$ near $u=0$. The convergence is faster if the rate of change of $\nabla p(u)$ is small and the rate of change of $c[\nabla\phi(u)]$ is large near $u=0$.

In Example 3, the rate of change of $\nabla\phi(u)$ is small near $u=0$ and convergence is slow, while in Example 4, it is large and convergence is rapid. In Example 5, the rate of change of $\nabla p(u)$ is small near $u=0$, thus the fast convergence.

The following example shows that in the absence of Assumption 1, Equation 2.18 may not lead to convergence for any $c>0$ when ϕ is essentially quadratic.

Example 6 [22]. Consider the problem, minimize $-|x|^{\rho}$ such that $x=0$ where $1<\rho<2$. For any $c>0$, there exists a neighborhood of $x=0$ such that the augmented Lagrangian (Eq. 2.14) does not have a local minimum for any λ when ϕ is essentially quadratic. This can be corrected by using $\phi(t)=|t|^{\rho'} + t^2/2$ where $1<\rho'<\rho$.

The two extreme cases are now examined. First, suppose $\bar{\lambda}^k = \bar{\lambda}^*$, Equation 2.19 becomes

$$|| \bar{X}^k - \bar{X}^* || \leq 0$$

The norm is always non-negative; therefore, $\bar{X}^k = \bar{X}^*$ and the solution is reached in one unconstrained minimization.

Next, letting $\bar{\lambda}^k = 0$ for all k, i.e., the penalty function method, Equation 2.19 becomes

$$|| \bar{X}^k - \bar{X}^* || < \frac{M || \bar{\lambda}^* ||}{c}$$

In this case, $\bar{X}^k \to \bar{X}^*$ only if $c^k \to \infty$, requiring many unconstrained minimizations and a sublinear rate of convergence.

It has been shown that the rate of convergence is directly dependent on the penalty parameter, c. The convergence estimates (Eqs. 2.19, 2.20) are valid for c greater than some

threshold value, c* which depends on ($\bar{\lambda}$) and the problem data. In general, c* is unavailable and it is impossible to know apriori the range of values of c for which Equations 2.19 and 2.20 are valid and imply fast convergence.

A penalty parameter update sequence is required to increase c^k monotonically with each unconstrained minimization. As $c \to \infty$, Equations 2.19 and 2.20 will eventually become valid. It should be noted that large values of c can induce ill-conditioning, making the unconstrained minimization of $A(\bar{X},\bar{\lambda},c)$ difficult. On the other hand, Equations 2.19 and 2.20 indicate faster convergence of $\{\bar{\lambda}^k\}$ to $\bar{\lambda}^*$ for large values of c.

An update sequence is recommended whereby c is multiplied by some constant $\gamma>1$, i.e., $c^{k+1} = \gamma c^k$. The penalty parameter c is increased in this manner to some significantly large c_{max}. Bertsekas [22] recommends γ not much larger than 1 to avoid ill-conditioning effects in the first few unconstrained minimization iterations. This update scheme will be used since it has been subject to the most testing in recent years. The choice of an initial c requires more experimentation and will be discussed later.

Other methods are available for updating the penalty parameter c. Powell [7] suggests multiplying c by some constant $\beta>1$ only if the violated constraint, as measured by $||h[\bar{X}(\bar{\lambda},c)]||$, is not decreased by a certain factor over the previous minimization, i.e., $c^{k+1} = \beta c^k$ if $||h[\bar{X}(\bar{\lambda}^k,c^k)]|| > \gamma||h[\bar{X}(\bar{\lambda}^{k-1},c^{k-1}]||$ and $c^{k-1} = c^k$ otherwise, where $\beta>1$, and, $\gamma<1$ are some specified

scalars. This scheme generates a penalty parameter sequence that will be constant after a certain index and will achieve convergence by virtue of enforcing the asymtotic feasibility of the constraints, i.e.,

$$\lim_{k\to\infty} ||h[\bar{X}(\bar{\lambda}^k, c^k)]|| = 0$$

Another similar update scheme is to use a penalty parameter vector such that c_i is updated by the Powell method only if the constraint $h_i[\bar{X}(\bar{\lambda},c)]$ is violated. This case of a separate penalty factor for each constraint corresponds to merely scaling of the constraints. A simple modification to Equations 2.19 and 2.20 is used to prove convergence of this case.

Finally, define a dual function $A_c(\bar{\lambda})$ for the augmented Lagrange function $A(\bar{X},\bar{\lambda},c)$ such that

$$A_c(\bar{\lambda}) = \min_{\bar{X}} A(\bar{X},\bar{\lambda},c) \tag{2.21}$$

It can be shown that if $X^k(\bar{\lambda},c)$ is the solution to Equation 2.21, then $A_c(\bar{\lambda})$ is a twice continuously differentiable convex function with gradient given by

$$\nabla_\lambda A_c(\bar{\lambda}) = \nabla h[\bar{X}^k(\bar{\lambda},c)] \tag{2.22}$$

where $h[\bar{X}^k(\bar{\lambda},c)]$ is the constraint vector (Eq. 2.1b) evaluated at X^k. Substituting Equation 2.22, Equation 2.18 becomes

$$\bar{\lambda}^{k+1} = \bar{\lambda}^k + c^k \bar{\nabla}_\lambda A_c(\bar{\lambda}) \tag{2.23}$$

This relation shows that the multiplier iteration (Eq. 2.18) is an iteration of steepest ascent for finding the maximum of the dual function, $A_c(\bar{\lambda})$. Equation 2.23 is equivalent to

a steepest ascent iteration for quadratic penalty functions, $\phi(t) = t^2/2$. The multiplier method can thus be viewed as a primal-dual method with a limited search for the optimal Lagrange multipliers in the dual space.

The Hestenes' multiplier method for nonlinear equality constrained problems will not be summarized. Quadratic penalty functions are used in this algorithm.

Step 1: Select $\bar{\lambda}^o=0$ and an appropriate penalty parameter $c^o>0$. Set k = 1.

Step 2: Solve min $A(\bar{X}, \bar{\lambda}^k, c)$, defined by Equation 2.13. Denote the solution $\bar{X}^k$.

Step 3: Update $\bar{\lambda}^k$ by Equation 2.18.

Step 4: If $\bar{\lambda}^{k+1} = \bar{\lambda}^k$, stop. $(\bar{X}^k, \bar{\lambda}^k)$ is the optimal solution. Otherwise, go to Step 5.

Step 5: Set $\bar{\lambda}^k = \bar{\lambda}^{k+1}$. If $c^k<c_{max}$, where c_{max} is some significantly large number, update c^k by $c^{k+1} = \gamma c^k$, where γ is some increase factor greater than one. Otherwise, $c^k=c_{max}$. Set k=k+1. Go to Step 2.

Note that no assumptions or restrictions have been made concerning the nature of the objective or constraint function. If Assumptions 1 and 2 are satisfied, the solution will converge to a global minimum; otherwise, only a local minimum can be guaranteed.

E. THE INEQUALITY CONSTRAINED PROBLEM

The multiplier method has been shown to be theoretically attractive for equality constrained minimization problems. In reality, many engineering problems involve inequality constraints. It is, therefore, necessary to extend the discussion of the multiplier method to include inequality constraints.

Consider the nonlinear inequality constrained problem:

$$\text{Minimize } f(\bar{X}) \tag{2.24a}$$

$$\text{such that } g_i(\bar{X}) \leq 0 \quad (i=1,\ldots,\ell) \tag{2.24b}$$

where $\bar{X}$ is the vector of n design variables. Introducing slack variables, Equation 2.24b becomes

$$g_i(\bar{X}) + z_i^2 = 0 \tag{2.25}$$

where z_i^2 is the slack variable for the i^{th} constraint. The problem is now an equality constrained problem of the form of Equation 2.1; however, the number of design variables has increased to $n+\ell$. The augmented Lagrangian (Eq. 2.13) becomes

$$A(\bar{X},\bar{Z},\bar{\lambda},c) = f(\bar{X}) + \sum_{i=1}^{m} \lambda_i (g_i(\bar{X}) + x_i^2) + \frac{c}{2} \sum_{i=1}^{\ell} (g_i(\bar{X}) + x_i^2)^2 \tag{2.26}$$

If the number of constraints, ℓ is much greater than the number of design variables, n, as is often the case in engineering design problems, the unconstrained minimization problem becomes sizable. The scope of the problem can, however, be reduced by eliminating the slack variables, z_i^2.

The unconstrained function, $A(\bar{X},\bar{Z},\bar{\lambda},c)$ is first minimized with respect to $\bar{Z}$. For a local minimum to exist, the stationary conditions

$$\frac{\partial A}{\partial z_i} = 0 \quad (i=1,\ldots,n) \tag{2.27}$$

must hold. Differentiating $A(\bar{X},\bar{Z},\bar{\lambda},c)$

$$\frac{\partial A}{\partial z_i} = 2\lambda_i x_i + c[g_i(\bar{X}) + z_i^2] \ (2z_i) = 0 \tag{2.28}$$

$$z_i[\frac{\lambda_i}{c} + g_i(\bar{X}) + z_i^2] = 0 \tag{2.29}$$

The solution to Equation 2.29 is

$$z_i^2 = 0$$

or

$$z_i^2 = -\frac{\lambda_i}{c} - g_i(\bar{X})$$

Since $z_i^2 < 0$ is meaningless, the solution becomes

$$z_i^2 = \max\ [0,\ -g_i(\bar{X}) - \frac{\lambda_i}{c}] \tag{2.30}$$

Equation 2.30 shows that z_i is no longer an independent variable. From this equation it is observed that if $g_i(\bar{X})$ is a critical constraint, $z_i = 0$. If $g_i(\bar{X})$ is non-critical, $z_i > 0$. Therefore,

$$g_i(\bar{X}) + z_i^2 = \max\ [g_i(\bar{X}),\ -\frac{\lambda_i}{c}] \tag{2.31}$$

With the slack variables eliminated, the augmented Lagrangian becomes

$$A(\bar{X},\bar{\lambda},c) = f(\bar{X}) + \sum_{i=1}^{\ell} [\lambda_i \psi_i + \frac{c}{2} \psi_i^2]$$

$$\text{where } \psi_i = \max\ [g_i(\bar{X}),\ -\frac{\lambda_i}{c}] \tag{2.32}$$

This is referred to as Rockafellar's augmented Lagrange function. Note that $A(\bar{X},\bar{\lambda},c)$ has continuous first derivatives with respect to c, but discontinuous second derivatives at $g_i(\bar{X}) = -\bar{\lambda}/c$. This preludes the use of second order methods, i.e., Newton's method, for unconstrained minimization.

From Equation 2.31, Hestenes' update formula for $\bar{\lambda}$ for the inequality constrained problem becomes

$$\lambda_i^{k+1} = \max\ [0,\ \lambda_i^k + c\ g_i(\bar{X}^k)] \qquad (2.33)$$

The algorithm for Hestenes' multiplier method is easily modified to include inequality constraints by adding Equations 2.32 and 2.33 to steps 2 and 3, respectively. Note that since the inequality constrained problem is transformed to an equivalent equality constrained problem, the convergence properties are identical.

Example 1 can now be solved by the multiplier method.

Example 7 [2]. Minimize x such that $1-x\leq 0$.

Solution: Equation 2.32 becomes

$$A(x,\lambda,c) = x + c\begin{cases} \frac{1}{2}(1-x)^2 + \frac{\lambda}{c}(1-x), & \text{if } 1-x\geq 0 \\ -\frac{\lambda^2}{2c^2}, & \text{otherwise} \end{cases}$$

Applying Hestenes' algorithm with $\lambda^o=0$ and $c^o=1$, the augmented Lagrangian becomes

$$A^o(x,0,1) = x + \frac{1}{2}\begin{cases} (1-x)^2, & \text{if } 1-x\geq 0 \\ 0, & \text{otherwise} \end{cases}$$

This has a solution $x^o=0$. Updating λ by Equation 2.32,

$$\lambda^1 = \max [0,\lambda^o+cg(x^1)]$$

$$= \max [0, 0 + 1(1)] = 1$$

With $\lambda^1=1$ and $c^1=1$, the next unconstrained minimization of $A(x,\lambda,c)$ becomes

$$A^1(x,1,1) = x + \begin{cases} (1-x)^2/2 + (1-x), & \text{if } 1-x \geq 0 \\ - 1/2 , & \text{otherwise} \end{cases}$$

The solution becomes $x' = 1$. Updating λ,

$$\lambda^2 = \max [0,\lambda^1 + c^1 g^1(x^1)]$$

$$= \max [0, 1 + 1(0)] = 1$$

Since $\lambda^1 = \lambda^2$, stop the calculation. The optimal solution is $(x^*,\lambda^*) = (1,1)$.

Note the convergence of the method even for a constant penalty parameter c.

The functions A^o and A^1 are plotted in Fig. 5, and the objective value vs. c is plotted in Fig. 6. Several facts are evident from this example.

1. A solution is obtained in a few unconstrained minimizations.

2. Each unconstrained problem is a smooth curve. Thus ill-conditioning is avoided.

3. Convergence is not asymtotic and an exact solution is attainable.

4. With an initial Lagrange multiplier of zero, the solution is obtained from the infeasible region.

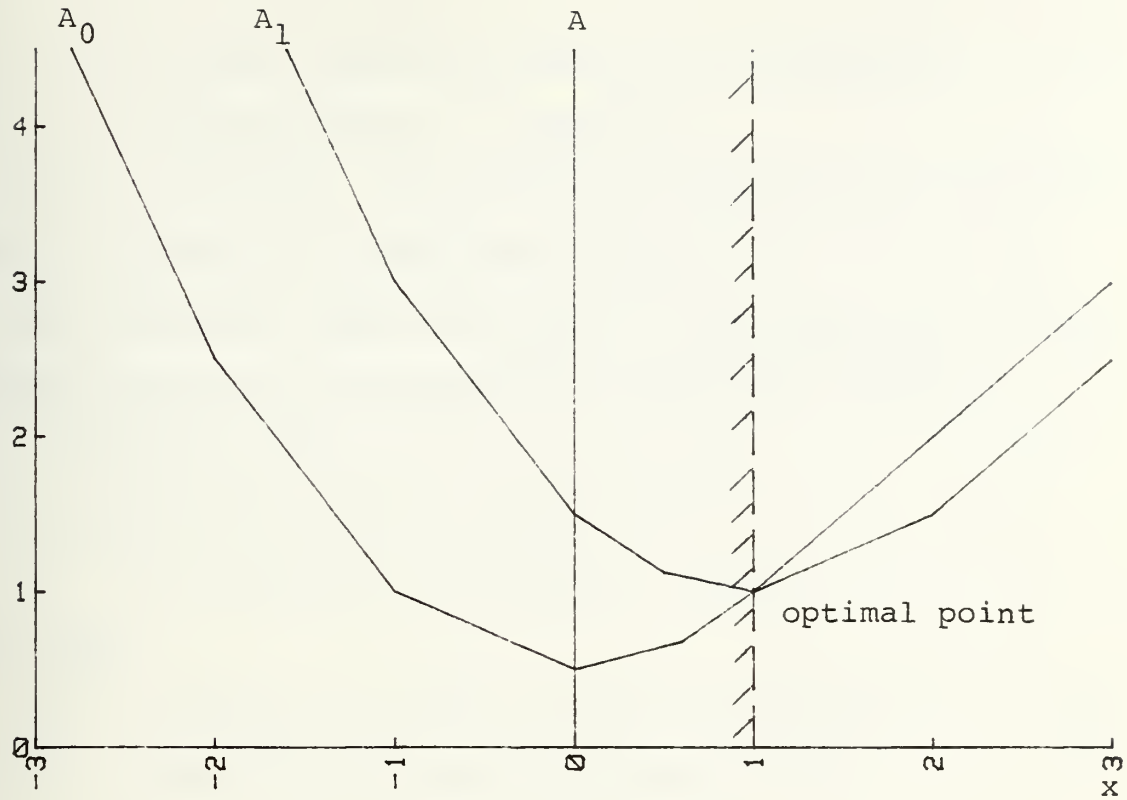


Fig. 5. Augmented Lagrangian Function

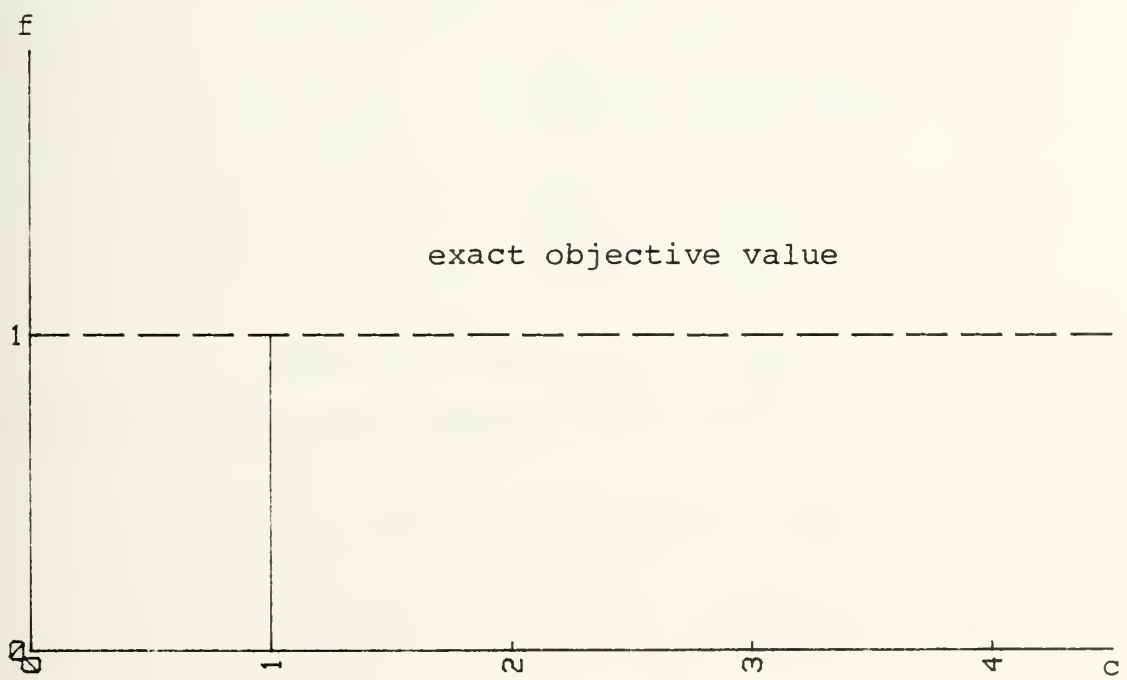


Fig. 6. Convergence of Augmented Lagrangian Function

5. The optimal Lagrange multiplier is obtained.

6. Any starting point is possible.

The method, therefore, shows attractive theoretical features over pure penalty or primal-dual methods.

The multiplier method algorithm is now summarized for the nonlinear equality and inequality constrained problem:

Minimize $f(\bar{X})$

such that $g_i(\bar{X}) \leq 0 \quad (i=1,\ldots,\ell)$

and $h_j(\bar{X}) = 0 \quad (j=\ell+1,\ldots,\ell+m)$

Algorithm:

Step 1. Choose an initial Lagrange multiplier $\bar{\lambda}^0 = 0$ and an initial penalty parameter $c^0 > 0$.

Step 2. Solve the unconstrained minimization problem

$$\text{Min } A(\bar{X},\bar{\lambda},c) = f(\bar{X}) + \sum_{i=1}^{m} [\lambda_i \psi_i + \frac{c}{2} \psi_i^2] + \sum_{j=\ell+1}^{m+\ell} [\lambda_j h_j(\bar{X}) + \frac{c}{2} h_j^2(\bar{X})]$$

where $\psi_i = \max [g_i(\bar{X}), -\frac{\lambda_i}{c}]$

Step 3. Update the Lagrange multipliers, $\bar{\lambda}$ by

$$\lambda_i^{k+1} = \lambda_i^k + c \quad \max[g_i(\bar{X}^k), -\frac{\lambda_i}{c}] \quad (i=1,\ldots,\ell)$$

$$\lambda_j^{k+1} = \lambda_j + c\, h_j(\bar{X}^k) \qquad (j=m+1,\ldots,m+\ell)$$

Step 4. If $\bar{\lambda}^k = \bar{\lambda}^{k+1}$, stop. The optimal solution is $(\bar{X}^*,\bar{\lambda}^*) = (\bar{X}^k,\bar{\lambda}^k)$. Otherwise, go to Step 5.

Step 5. Set $\bar{\lambda}^k = \bar{\lambda}^{k+1}$. Update the penalty parameter c by $c^{k+1} = \gamma c^k$.

If $c^{k+1} \geq c_{max}$, set $c^k = c_{max}$, else set $c^k = c^{k+1}$.

Set k = k+1; go to Step 2.

This is the theoretical algorithm used in the multiplier method. Minor variations are described in the next chapter which are designed to improve the computational efficiency, reliability, and accuracy of the method.

III. COMPUTATIONAL ASPECTS

The effectiveness of the multiplier method is dependent upon the accuracy of the unconstrained minimization, choice of Lagrange multiplier update formula, penalty parameter, the form of the penalty function, and the objective and constraint functions themselves. The theoretical effect of the Lagrange multiplier and penalty parameters on the method's convergence rate was described in Chapter II. The requirement that the objective and the constraint functions satisfy Assumptions 1 and 2, listed in Chapter II.D, has also been established.

This chapter will examine the effects of the Lagrange multiplier and penalty parameter from a computational viewpoint. As stated earlier, the penalty function is quadratic throughout this discussion. The computational aspects of the unconstrained minimization program and associated one-dimensional search subprograms are first examined.

A. THE UNCONSTRAINED MINIMIZATION PROBLEM

A sequence of unconstrained minimizations is required in obtaining a solution by the multiplier method. Of the many unconstrained minimization techniques available, the Davidon-Fletcher-Powell [DFP] method [1] was selected because of its accuracy and rapid convergence. Figure 7 compares the DFP

COMPARISON OF UNCONSTRAINED MINIMIZATION METHODS

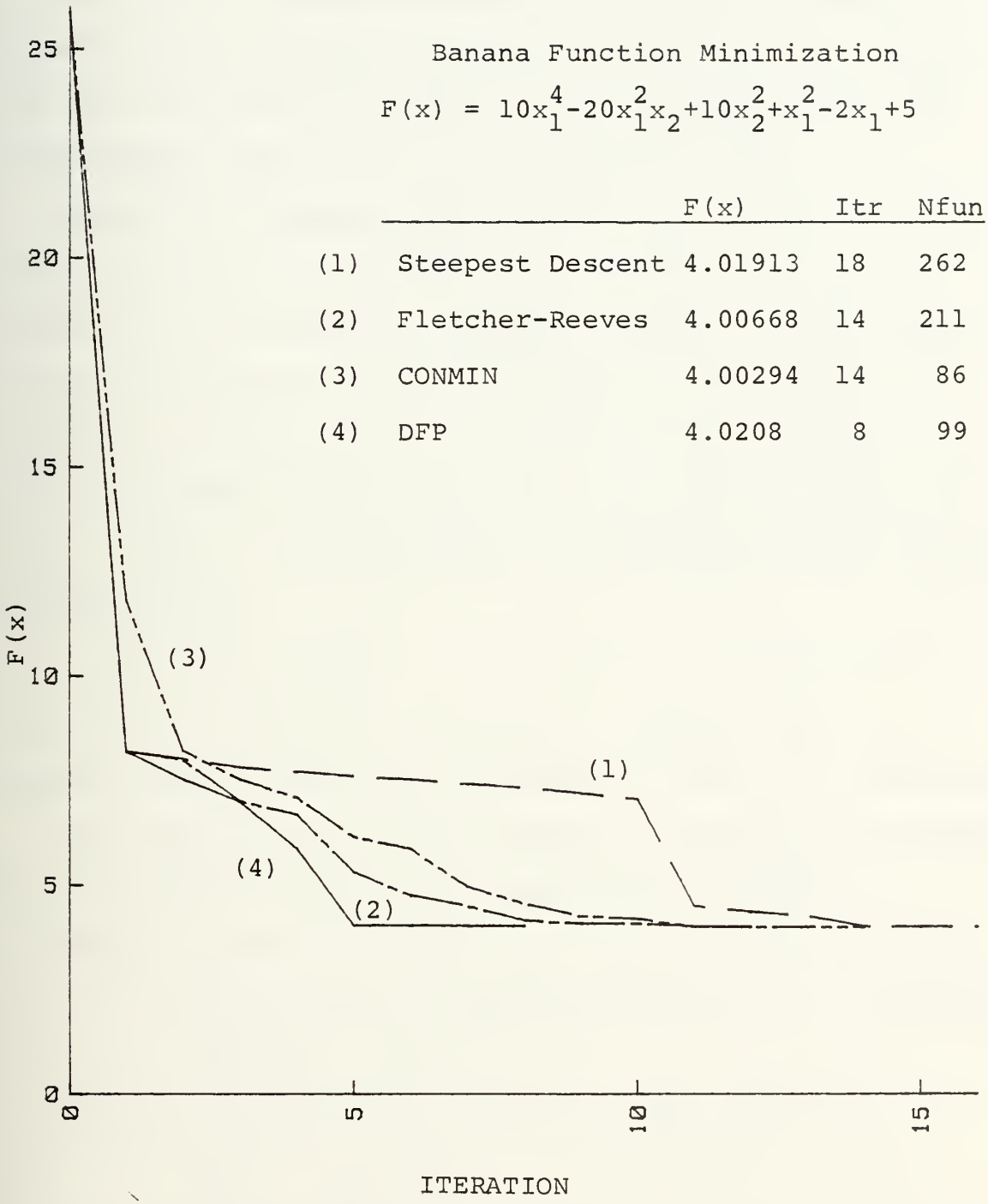


Fig. 7. Comparison of Unconstrained Minimization Methods

method to the methods of steepest descent, Fletcher-Reeves conjugate gradient method developed as part of this study, and the Fletcher-Reeves method contained in the CONMIN program for the extremely ill-conditioned "banana" function. Note the DFP method converges in fewer iterations than the other methods with little or no loss of accuracy.

To solve the general unconstrained minimization problem, it is necessary to first calculate a search direction, $\bar{S}$, initially the direction of steepest descent. A one-dimensional search is then used to find the minimum of the objective function in the search direction, $\bar{S}$. The one-dimensional search computes a scalar, α_k^* which minimizes the function $f(\bar{X}_{k-1} + \alpha_k \bar{S}_k)$. The new X-vector associated with the function minimum becomes

$$\bar{X}_k = \bar{X}_{k-1} + \alpha_k \bar{S}_k$$

A new search direction is calculated, and the iteration is repeated until a specified convergence criteria is satisfied.

The algorithm for the DFP method is described as follows for the general unconstrained minimization problem:

Step 1. Select an initial point $\bar{X}_o$ and an initial positive definite symmetric matrix, $H_o = I$ (identity matrix). Set k=1.

Step 2. Calculate the gradient, $\nabla f(\bar{X}_o)$, and set

$$\bar{S}_o = - H_o \nabla f(\bar{X}_o)$$

Step 3. Compute $\bar{X}_k = \bar{X}_{k-1} + \alpha_k^* \bar{S}_k$

where α_k^* minimizes $f(\bar{X}_{k-1} + \alpha\bar{S}_k)$, i.e., α_k^* is the solution to the one-dimensional search.

Step 4. Computer $H_{k+1} = H_k + \bar{A}_k - \bar{B}_k$

where $\Delta\bar{F} = \nabla f(\bar{X}_k) - \nabla f(\bar{X}_{k-1})$

$$\Delta\bar{X} = \bar{X}_k - \bar{X}_{k-1} = -\alpha^*\bar{S}_k$$

$$\bar{A}_k = \frac{\Delta\bar{X}\Delta\bar{X}^T}{\Delta\bar{X}^T\Delta\bar{F}}$$

$$\bar{B}_k = \frac{(H\Delta\bar{F})(H\Delta\bar{F})^T}{\Delta\bar{F}^T H\Delta\bar{F}}$$

Step 5: Compute $\bar{S}_{k+1} = -H_{k+1}\nabla f_k$, set k=k+1, go to Step 3. The basic algorithm is modified to provide (1) scaling of the variables, (2) appropriate update and resets, and (3) a suitable termination criterion.

A feature is added which normalizes the search vector such that

$$S_{k_{NORM}} = \frac{\bar{S}_k}{|\bar{S}_k|_{max}}$$

For excessively ill-conditioned problems, pre-scaling the variables may also be required.

Resets and updates of the vector, $\bar{S}$ and the matrix, H, are incorporated to prevent method breakdown from round-off error and other instabilities. The positive definiteness of H is preserved in theory only if α_k^* provides a true minimum point, i.e., $\nabla\bar{F}_{k+1}^T \bar{S}_k = 0$. The update of H is, therefore, skipped if $\nabla\bar{F}_{k+1}^T \bar{S}_k > \varepsilon$, $H \neq I$, and the update

was not skipped on the previous iteration. A tolerance value, $\varepsilon = 0.01$ was co-putationally found to provide a good update criteria with negligible effect on the convergence side.

Two checks were made on $\bar{S}$ to insure a valid search direction. Note first that if H=I, the search direction is that of steepest descent. Now at iteration k, if $\bar{S}_k \cdot \nabla f_k(\bar{X}) > 0$, the search direction will not reduce the objective and a new search direction is found. Second, if at the end of the one-dimensional search $f(\bar{X}_{k+1}) > f(\bar{X}_k)$, the function is obviously not a minimum and a new search direction is found, In each case, the H-matrix is reset to identify, I, and a search in the direction of steepest descent is performed. Finally, to maintain stability of the H-matrix, it is reset to the identify matrix every NDV+1 iterations, where NDV is the number of design variables.

The method is terminated when the relative or absolute difference of the objective function is less than 0.001 for two consecutive iterations or when a preassigned maximum number of iterations is exceeded. The somewhat strict convergence criterion was chosen since the performance of the multiplier method is highly dependent on an accurate unconstrained minimization.

The DFP method has been shown to be a reliable first order method with quadratic convergence. It has one

additional feature in that it provides a good approximation to the Hessian matrix,

$$H \simeq \frac{\partial^2 f(\bar{X})}{\partial \bar{X}^2}$$

Thus, the method provides many of the advantages of a second order, Newton type method, without the tedious calculation of second derivatives.

Finally, the DFP method requires subprograms to calculate gradients and perform the one-dimensional search. Gradient calculation was done by the first forward finite difference method. While a central difference method would improve the accuracy of the calculations, the first forward finite difference method with a step size of 0.01 has provided acceptable results in experimentation thus far. Discussion of the one-dimensional search subprogram follows in the next section.

B. THE ONE-DIMENSIONAL SEARCH

A combination of the Golden Section method and polynomial interpolation was used for the one-dimensional search. The one-dimensional search calculates a scalar, α_k^* which minimizes the function, $f(\bar{X}_{k-1} + \alpha_k \bar{S}_k)$, It has been shown that the accuracy of computing α_k is critical to the performance of the unconstrained minimization subprogram. The Golden Section method was used because of its reliability and accuracy. The method is, however, less efficient than polynomial interpolation methods in many cases. The Golden Section method

converges linearly while polynomial methods have no guaranteed rate of convergence. It may also require many function evaluations per iteration while polynomial methods require few.

The Golden Section method is based on a Fibonacci search [16] which yields a sequence of intervals of uncertainty whose widths tend to zero faster than any other method. Given upper and lower bounds, x_u and x_ℓ, where x denotes the scalar, α, in Equation 3.1, two interior points, x_1 and x_2 are found such that

$$x_u - x_2 = x_1 - x_\ell$$

and the ratio

$$\frac{x_2}{x_1} = (1 + \sqrt{5})/2 = 1.61803$$

This is the Golden Section number. The interior points, x_1 and x_2 are defined as

$$x_1 = (1 - \tau)x_L + \tau x_u$$

$$x_2 = \tau x_\ell + (1 - \tau)x_u$$

$$\text{where } \tau = (3-\sqrt{5})/2 = 0.38197$$

The bounds are revised by a simple update routine and one new interior point is calculated at each step. The process is continued until a given relative convergence tolerance, ε, is satisfied. The algorithm for the Golden Section search is given in Appendix A.

The relative convergence tolerance, ε, is defined

$$\varepsilon = \Delta x/(x_u - x_\ell)$$

Recognizing that the interval is reduced by the fraction, τ (38 percent) each iteration, the maximum number of function evaluations, N per iteration is calculated

$$\varepsilon = (1-\tau)^{N-3}$$

or $$N = \ln(\varepsilon)/\ln(1-\tau) + 3$$

$$= -2.078 \ln(\varepsilon) + 3$$

Note that by defining ε, a fixed number of function evaluations are performed each iteration. If the absolute convergence tolerance, Δx were specified, the number of function evaluations will vary each iteration. Experiments have shown an absolute convergence criteria to yield slightly better accuracy with nearly the same number of function evaluations. Therefore, an absolute convergence criteria was specified to give the same or better efficiency than the relative criteria and with improved accuracy.

The Golden Section method requires a logical guess to the first x_u such that x_u and x_ℓ bracket the minimum. Selection of x_u was performed such that if more than two guesses were required, the last three points chosen would yield the values x_ℓ, x_1, and x_u. This provided a smaller initial bound than if the original x_ℓ was used and also saved one function evaluation since x_1 was already found for future calculations.

Finally, a cubic approximation was performed using the last four points from the Golden Section method. A better

minimum was found in almost every case at the cost of only one additional function evaluation.

C. THE MULTIPLIER METHOD

Experimentation with the multiplier method involved investigating the effects of the penalty parameter, c, the update factor, γ, and the update formula for the multipliers, $\bar{\lambda}$. It was desired to compute these parameters internally if possible, such that the method's performance would be independent of the problem. This is necessary to provide a method which can be used simply and reliably for many engineering applications. Effects of scaling, variable bounds, and perturbed constraints were not investigated but will be discussed briefly for completeness.

The choice of the initial penalty parameter, c^o, can have significant effect on the efficiency of the multiplier method. No universally acceptable method has been found to select an initial penalty parameter, c^o. Two methods have been found to provide good initial estimates and are recommended here: (1) choose the initial penalty parameter, c^o, of the same order ot magnitude as the initial objective function, or (2) choose the initial penalty parameter such that $|\nabla f|=c|\nabla g_{max}|$. The second method requires the gradient of each constraint which is not directly available in the current program. Both methods have provided acceptable results for the examples tested. They cannot guarantee a reasonable convergence or a reliable answer in all cases.

Bertsekas [22] recommends as increase factor, γ not much greater than 1. A γ=2 worked well in all the cases tested. Reasonable success was achieved with γ=5, but caution should be used since instability and ill-conditioning occurred when c^o was chosen too large. Bertsekas' recommendation of $\gamma>1$ to a maximum of γ=2 appears to be adequate in most cases.

The maximum penalty parameter, c_{max} is not as critical to problem stability as the initial c^o. It was noted that ill-conditioning does occur for c_{max} too large. A good guideline is to choose c_{max} of order five to six times that of c^o. If ill-conditioning occurs, it will be necessary to lower c_{max}, c^o or both.

Choosing an initial set of multipliers, $\bar{\lambda}^o$ other than zero was not investigated. The convenience of identifying active and violated constraints without additional computations when λ_i^o=0 tends to overshadow any computational advantage which may be attained by choosing a non-zero initial multiplier.

Normalization of the constraints is necessary in the function subprogram to avoid breakdown of the multiplier method. Scaling may be required for extremely ill-conditioning problems. Various scaling methods exist, all of which provide some means making the variables, objective, and constraints (or their gradients) equal to or near the same order of magnitude.

At present, the algorithm treats all side constraints or bounds as additional problem constraints. Treating side constraints directly can greatly improve the computational efficiency, as well as simplifying problem programming. Ragsdell [23] has shown success in treating variable bounds (side constraints) directly but does not offer any theoretical justification. Further research on direct treatment of side constraints is required.

The multiplier method often converges computationally to a solution that is slightly infeasible. This may not be desirable from the practical point of view. Imai [2] considered the following perturbed problem:

Minimize $f(\bar{X},\bar{\lambda})$

such that $g_i(\bar{X},\bar{\lambda}) \leq -\varepsilon$ $(\varepsilon>0,\ i=1,\ldots,m)$

where ε is some small positive number. The constraints are pushed slightly into the feasible region, so the function will terminate in the feasible region.

The augmented Lagrangian, L becomes

$$L = f + c\sum_{i=1}^{m}\left[\lambda_i\psi_i + \frac{c}{2}\psi_i^2\right] \qquad (3.2)$$

where $\psi_i = \max\left[g_i + \varepsilon, -\frac{\lambda_i}{c}\right]$

Setting $\varepsilon=0$, Equation 3.2 reduces to the original Lagrange function for the general inequality constrained problem. Solutions obtained with $\varepsilon=0$ are only slightly infeasible, i.e., $g_i \simeq +10^{-3}$; therefore, accuracy of the original Lagrange function was acceptable even though slightly infeasible.

Finally, a termination criteria was chosen such that the relative of absolute change in the objective function was less than 0.001 for three consecutive iterations, or if a preassigned number of iterations was exceeded. This was chosen since convergence of the objective function is of more interest than convergence of the multipliers, $\bar{\lambda}$.

IV. COMPUTATIONAL EXPERIENCE

The multiplier method was used to solve five test cases. Three mathematical examples were used to demonstrate the method's computational performance as compared to other methods. Two engineering examples were then solved to show the application of the multiplier method in engineering design optimization. Results of the multiplier method computations are given in Table I. Table II shows a comparison of the multiplier method to other optimization methods for each case.

A. CASE 1: THE CONSTRAINED ROSEN-SUZUKI FUNCTION [5]

$$\text{Minimize } f(x) = x_1^2-5x_1+x_2^2-5x_2+2x_3^2-21x_3+x_4^2+7x_4+50$$

$$\text{such that } g_1(x) = x_1^2+x_1+x_2^2-x_2+x_3^2+x_3+x_4^2-x_4-8 = 0$$

$$g_2(x) = x_1^2-x_1+2x_2^2+x_3^2+2x_4^2-x_4-10 \leq 0$$

$$g_3(x) = 2x_1^2+2x_1+x_2^2-x_2+x_3^2-x_4-5 = 0$$

This problem was solved for two individual cases. Case 1A solves the problem with constraints $g_1(x)$ and $g_3(x)$ as equality constraints. Case 1B treats these two constraints as inequalities. The solution to each case is given in Table I. This problem demonstrates the method's ability to solve equality constrained problems directly with greater accuracy and efficiency. As seen in Table II, the method performs

no better than the exterior penalty method or CONMIN for the inequality constrained problem. However, when the equality constraints are treated directly, the multiplier method shows significant improvement. It should be noted that CONMIN uses a polynomial interpolation one-dimensional search, which is significantly more efficient than the Golden Section search in many cases. Using the polynomial search should make the multiplier method more efficient, if reliability can be preserved. The reliability of the Golden Section search made it the more desirable choice in the development of the method.

B. CASE 2. A SIMPLE QUADRATIC FUNCTION [23]

Minimize $f(x) = 4x_1 - x_2^2 - 12$

such that $g_1(x) = 25 - x_1^2 - x_2^2 = 0$

$g_2(x) = -10x_1 + x_1^2 - 10x_2 + x_2^2 + 34 \leq 0$

$g_3(x) = -x_1 \leq 0$

$g_4(x) = -x_2 \leq 0$

This case was chosen as a comparison to the Sequential Unconstrained Minimization Technique, SUMT [23], using an interior penalty function method. The problem solution is given in Table I. From Table II, it can be seen that the multiplier method is comparatively accurate to the SUMT method.

C. CASE 3: THE PAVIANI FUNCTION [23]

Minimize $f(x) = 1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3$

such that $g_1(x) = x_1^2+x_2^2+x_3^2-25 = 0$

$g_2(x) = 8x_1+4x_2+7x_3-56 = 0$

$g_3(x) = -x_1 \leq 0$

$g_4(x) = -x_2 \leq 0$

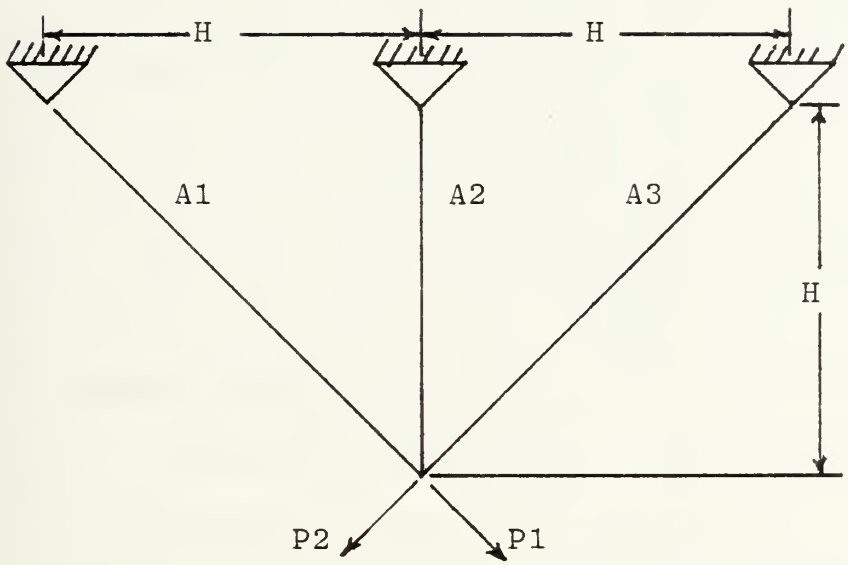
$g_5(x) = -x_3 \leq 0$

In this case, it is interesting to note that even though the accuracy and efficiency is comparable, the penalty method required a larger penalty parameter, c than the multiplier method. This is a common point and shows the multiplier method's ability to avoid ill-conditioning by efficiently and accurately obtaining a solution with a finite c.

D. CASE 4: THE THREE BAR TRUSS PROBLEM

As a simple structural design problem, the three bar truss in Fig. 8 was considered. The problem was to determine the areas A1, A2, A3 to minimize the structure weight, W. The design was subject to constraints $-15000 < \sigma_{ij} < 20000$ psi where σ_{ij} is the stress in the truss member, i under load condition, j. An additional geometric constraint of A1=A3 was imposed to maintain symmetry.

THE THREE BAR TRUSS PROBLEM



MATERIAL: $\rho = 0.1$ lb/in^3

$E = 10^6$ psi

GEOMETRY: H = 10 inches

LOADS: $P_1 = P_2 = 20{,}000$ lbs.

STRESS LIMITS: $-15{,}000 \leq \sigma_{ij} \leq 20{,}000$ psi

INITIAL AREAS: A1 = A2 = A3 = 1.0 sq.in.

FIGURE 8.

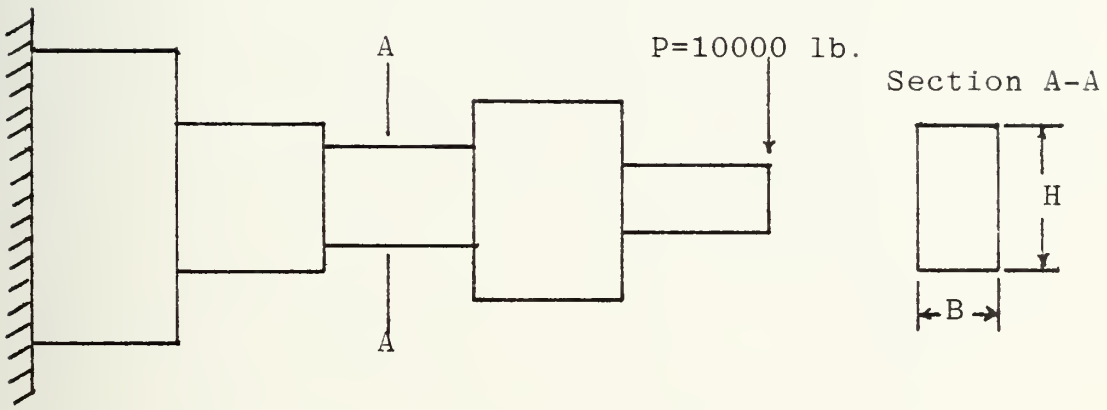
E. CASE 5: CANTILEVER BEAM PROBLEM

A cantilever beam with base, B, and height, H, variable over the length, L, was optimized to find the minimum volume. The beam, as seen in Fig. 9, was divided into five sections. The following constraints were imposed: (1) bending stress, σ in each section was not to exceed ± 20000 psi. (2) deflection, δ under the load was not to exceed ± 2.0 inches, and (3) height to beam ratio, $(H/B) < 30$. Additional side constraints were also imposed on the base and height. Separating the beam into five sections expanded the problem to one of ten design variables and 37 inequality constraints. The problem was solved by the multiplier method with $c=1000$ and $\gamma=1$. The solution is given in Table I, with a comparison to the exterior penalty method and CONMIN in Table II. It should be noted that the exterior penalty method did not obtain a solution due to numerical ill-conditioning and stability problems.

F. SUMMARY

For the cases involving equality constraints, the advantages of the multiplier method over penalty methods and CONMIN has been shown. The multiplier method has proven to be a suitable alternative to the other methods tested for both equality and inequality constrained problems. While the method may have worked well for the cases tested, it cannot be guaranteed to be the most efficient, if even the most accurate, in all cases. The dependence of the multiplier

THE CANTILEVER BEAM PROBLEM



The beam is broken into five segments of equal length and each segment is governed by two design variables for a total of ten design variables.

Design requirements:

MATERIAL: $E = 30 \times 10^6$ psi

GEOMETRY: Total length, L = 200 inches

$1 \leq H_i \leq 30$ inches $\quad i = 1,5$

$0.5 \leq B_i \leq 5$ inches $\quad i = 1,5$

$H_i/B_i \leq 30 \quad i = 1,5$

STRESS LIMITS: (at the left end of each segment)

$$\sigma_i = \frac{Mc}{I} \leq 20000 \text{ psi} \quad i = 1,5$$

FIGURE 9

method on internal programming parameters, c and λ, may even make the method less attractive in some cases to other optimization methods.

TABLE I

MULTIPLIER METHOD RESULTS

Case 1A: The Equality Constrained Rosen-Suzuki Function

Initial Obj = 31.000

$\bar{X}^T$ = (1.0, 1.0, 1.0, 1.0)

$\bar{G}^T$ = (-4.0, -6.0, -1.0)

Final Obj = (6.0075

$\bar{X}^T$ = (0.01607, 1.0285, 1.9799, -1.018)

$\bar{G}^T$ = (-0.00515, -0.8896, 0.004387)

Theoretical Optimum

Obj = 6.0000

$\bar{X}^T$ = (0.0, 1.0, 2.0, -1.0)

$\bar{G}^T$ = (0.0, -1.0, 0.0)

Case 1B: The Inequality Constrained Rosen-Suzuki Function

Initial Obj = 31.000

$\bar{X}^T$ = (1.0, 1.0, 1.0, 1.0)

$\bar{G}^T$ = (-4.0, -0.0, -1.0)

Final Obj = 6.1178

$\bar{X}^T$ = (0.03206, 1.0334, 1.9509, -1.054)

$\bar{G}^T$ = (-0.1166, -0.8921, -0.1237)

Theoretical Optimum = Same as above

TABLE I (contd)

Case 2: A Simple Quadratic Function

Initial Obj = -9.0

$\bar{X}^T$ = (1.0, 1.0)

$\bar{G}^T$ = (23.0, 16.0, -1.0, -1.0)

Final Obj = -31.989

$\bar{X}^T$ = (1.0019, 4.8986)

$\bar{G}^T$ = (-0.0017, -0.192, -1.026, -4.894)

Theoretical Optimum

Obj = -32.000

$\bar{X}^T$ = (1.0, 4.8990)

$\bar{G}^T$ = (0.0, 0.0102, -1.0, -4.899)

Case 3: The Paviani Function

Initial Obj = 976.0

$\bar{X}^T$ = (210, 2.0, 2.0)

$\bar{G}^T$ = (-13.0, 2.0, -2.0, -2.0, -2.0)

Final Obj = 961.79

$\bar{X}^T$ = (3.289, 0.2403, 3.7599)

$\bar{G}^T$ = (0.0035, -0.0048, -3.286, -0.241, -3.761)

Theoretical Optimum

Obj = 961.715

$\bar{X}^T$ = (3.512, 0.217, 3.552)

$\bar{G}^T$ = (0.0, 0.0, -3.512, -0.217, -3.552)

TABLE I (contd)

Case 4: The Three Bar Truss

Initial Obj = 3.8284 lbs.

$\bar{X}^T$ = (1.0, 1.0, 1.0) in.

σ_{11} = 14142.2 psi

σ_{21} = 8284.2

σ_{31} = -5858.0

σ_{12} = -5858.0

σ_{22} = 8284.2

σ_{32} = 14142.2

Final Obj = 2.639

$\bar{X}^T$ = (0.7885, 0.4086, 0.7883)

σ_{11} = 20001.

σ_{21} = 14636.

σ_{31} = -5364.8

σ_{21} = -5364.8

σ_{22} = 14636.

σ_{32} = 20005.

Theoretical Optimum

Obj = 2.632

$\bar{X}^T$ = (0.781, 0.424, 0.781)

σ_{11} and σ_{32} are active constraints.

TABLE I (contd)

Case 5: Cantilever Beam

Initial Obj = 9000 cu. ft.

$\bar{H}^T$ = (15.0, 15.0, 15.0, 15.0, 15.0)

$\bar{B}^T$ = (3.0, 3.0, 3.0, 3.0, 3.0)

$\bar{\sigma}^T$ = (17778, 14222, 10667, 7111.2, 3555.6)

$(H/B)^T$ = (5.0, 5.0, 5.0, 5.0, 5.0)

δ = 1.0535

Final Obj = 3206.2 cu. ft.

$\bar{H}^T$ = (26.05, 24.62, 22.70, 19.83, 15.35)

$\bar{B}^T$ = (0.886, 0.796, 0.733, 0.646, 0.506)

$\bar{\sigma}^T$ = (19961, 19904, 19071, 18914, 20111)

$(H/B)^T$ = (29.42, 30.92, 30.97, 30.72, 30.32)

δ = 0.9528

TABLE II

COMPARISON OF VARIOUS OPTIMIZATION METHODS

	1A	1B	2	3	4	5
Number of Design Variables	4	4	2	3	4	5
Number of Equality Constraints	2	0	1	2	1	10
Number of Inequality Constraints	1	3	3	3	12	36
AUGMENTED LAGRANGE MULTIPLIER METHOD						
Number of Iterations	6	9	5	4	6	25
Number of Function Evaluatons	304	347	198	120	185	3390
Objective Function Value	6.0075	6.1178	-31.989	961.79	2.6385	3206.2
EXTERIOR PENALTY FUNCTION METHOD						
Number of Iterations	7	7	5	4	10	(2)
Number of Function Evaluations	414	319	213	120	310	
Objective Function Value	6.0065	6.0165	-31.964	961.79	2.6390	
CONSTRAINED FUNCTION MINIMIZATION (CONMIN)						
Number of Iterations	(1)	11		(1)	8	20
Number of Function Evaluations		68			36	245
Objective Function Value		6.0183			2.7204	3227.5
INTERIOR PENALTY FUNCTION METHOD (SUMT)						
Number of Iterations			8			
Number of Function Evaluations			68			
Objective Function Value			-31.990	961.72		

(1) Cannot be solved by this method.
(2) Breaks down due to ill-condition.

V. CONCLUSIONS

The multiplier method has been shown to be an accurate and efficient method for solving problems in engineering design optimization. In the particular cases tested, it showed comparable or improved performance to other optimization methods. In using a finite penalty parameter, c^o, the numerical ill-conditioning of penalty methods is avoided in most cases. The convergence of the multiplier method is at least linear while penalty methods converge asymtotically. Exact solutions are also attainable by the multiplier method. The ability to handle equality constraints directly makes it an attractive alternative to CONMIN for the equality constrained problem.

The multiplier method has other advantages which make it attractive. It can be used as an interior or exterior optimization method, i.e., the optimum can be approached from the feasible or infeasible region. Any reasonable starting point can be used. The dynamic selection of active constraints when $\bar{\lambda}^o = 0$ is also a feature of the multiplier method.

Research of multiplier methods is far from complete. Its use in the various disciplines of engineering design are endless. A multiplier algorithm needs to be developed which is independent of internal computational parameters. The algorithm needs to be streamlined for easy application by the

practicing engineer. The specific applications for which the multiplier method is most attractive need to be identified.

The Augmented Lagrange Multiplier method is an extremely useful program in computer-aided engineering design. Its applications today are few, but its possibilities are endless and invaluable.

LIST OF REFERENCES

1. Fox, R.L., Optimization Methods for Engineering Design, Addison-Wesley, 1971.

2. Imai, K., Configuration Optimization of Trusses by the Multiplier Method, Ph.D. Thesis, University of California at Los Angeles, June 1978.

3. Fax, D.H., and Mills, R.R., Jr., Transactions of ASME, V. 79, pp. 653-661, 1957.

4. Hedderich, C.P., Heat Exchanger Optimization, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1980.

5. Vanderplaats, G.N., NASA Ames Research Center Technical Memorandum NASA TM X-62,282, CONMIN - A FORTRAN Program for Constrained Function Minimization User's Manual, August 1973.

6. Hestenes, M.R., Optimization Theory, Wiley, 1975.

7. Hestenes, M.R., "Multiplier and Gradient Methods," Journal of Optimization Theory and Applications (JOTA), V. 4, No. 5, pp. 303-320, 1969.

8. Fletcher, R., Optimization, pp. 283-298, Academic Press, 1969.

9. Miele, A., and others, "Use of the Augmented Penalty Function in Mathematical Programming Problems, Part I and II, JOTA, V. 8, No. 2, pp. 115-153, 1971.

10. Miele, A., and others, "On the Method of Multipliers for Mathematical Programming Problems," JOTA, V. 10, No. 1, pp. 1-33, 1972.

11. Tripathi, S.S., and Navendra, K.S., "Constrained Optimization Problems Using Multiplier Methods," JOTA, V. 9, No. 1, pp. 59-70, 1972.

12. Rockafellar, R.T., "Augmented Lagrange Multiplier Functions and Duality in Non-Convex Programming," SIAM J. of Control, V. 12, No. 2, pp. 268-285, May 1974.

13. Rockafellar, R.T., "The Multiplier Method of Hestenes and Powell Applied to Convex Programming," JOTA, V. 12, No. 6, pp. 555-562, 1973.

14. Bertsekas, D.P., "Combined Primal-Dual and Penalty Methods for Con-trained Minimization," SIAM J. of Control, V. 13, No. 3, pp. 521-544, May 1975.

15. Bertsekas, D.P., "On Penalty and Multiplier Methods for Constrained Minimization," SIAM J. of Control and Optimization, V. 14, No. 2, pp. 216-235, February 1976.

16. Luenberger, D.G., Introduction to Non-Linear Programming, Addison-Wesley, 1973.

17. Schmit, L.A. and Fleury, C., "An Improved Analysis/ Synthesis Capability Based on Dual Methods-Access 3," AIAA 20th Structures, Structural Dynamics, and Materials Conference, pp. 23-50, April 1979.

18. Pierre, D.A., and Lowe, M.J., Mathematical Programming via Augmented Lagrangians, Addison-Wesley, 1975.

19. Haftka, R.T. and Starnes, J.H., "Application of a Quadratic Extended Interior Penalty Function for Structural Optimization," AIAA Journal, V. 14, No. 6, pp. 718-724, June 1976.

20. Buys, J.D., Dual Algorithms for Constrained Optimization, Ph.D. Thesis, Rijksuniversiteit de Leiden, 1972.

21. Rupp, R.D., "Approximation of the Classical Isoperimetric Problem, JOTA, V. 9, pp. 251-264, 1972.

22. Bertsekas, D.P., "Multiplier Methods: A Survey," Automatica, V. 12, pp. 133-145, 1976.

23. Ragsdell, K.M., and Root, R.R., "Computational Enhancements to the Method of Multipliers," TRANS. ASME, J. of Mechanical Design, pp. 517-523, July 1980.

24. Himmelblau, D.M., Applied Nonlinear Programming, McGraw-Hill, 1972.

APPENDIX A

ALGORITHM FOR GOLDEN SECTION ONE-DIMENSIONAL SEARCH

The algorithm for the one-dimensional search subprogram is outlined in this appendix. The one-dimensional search is first performed by the Golden Section method. A cubic approximation is then performed on the last four values found by Golden Section method. This approach proved to be very effective in obtaining an accurate solution. The one-dimensional search is performed as follows:

Step 1. Specify the initial interval x_ℓ and x_u, where x denotes the scalar, α, in Equation 3.1. Evaluate the functions $y_\ell = f(x_\ell)$, $y_u = f(x_u)$.

Step 2. Specify the absolute convergence tolerance, Δx. Calculate the relative convergence tolerance, ε, and the number of function evaluations, N, from the equations.

$$\varepsilon = \Delta x/(x_u - x_\ell)$$

$$N = -2.078 \ln(\varepsilon) + 3$$

The value N is calculated in floating point arithmetic and rounded off to the next higher integer. If $N<4$, set $N=4$.

Step 3. Calculate x_1 and x_2 by the equations

$$x_1 = (1 - \tau)\, x_\ell + \tau x_u$$

$$x_2 = \tau x_\ell + (1 - \tau)\, x_u$$

where $\tau = (3-\sqrt{5})/2 = 0.38197$

Evaluate $y_1 = f(x_1)$ and $y_2 = f(x_2)$.

Step 4: Set counter, K=4. This is because four functions have already been evaluated. If N=4, go to Step 9.

Step 5: If y_2 is greater than y_1, go to Step 7.

Step 6: y_1 is greater than or equal to y_2. x_1 is a new lower bound. Set:

$$x_\ell = x_1$$
$$y_\ell = y_1$$
$$x_1 = y_2$$
$$y_1 = y_2$$
$$x_2 = \tau x_\ell + (1-\tau)x_u$$
$$y_2 = f(x_2)$$
$$K = K + 1$$

Go to Step 8.

Step 7: Y_z is greater than y_1. x_2 is a new upper bound. Set:

$$x_u = x_2$$
$$y_u = y_2$$
$$x_2 = x_1$$
$$y_2 = y_1$$
$$x_1 = (1-\tau)x_\ell + \tau x_u$$
$$y_1 = f(x_1)$$
$$K = K + 1$$

Step 8: Check convergence. If K>N, go to Step 9; else go to Step 5.

Step 9: Do cubic approximation with values x_ℓ, x_1, x_2, x_u.

$$y_c = f(x_\ell, x_1, x_2, x_u)$$

Step 10: Pick best of y_ℓ, y_1, y_2, y_c.

$Y = \min (y_\ell, y_1, y_2, y_c)$

X = corresponding x_ℓ, x_1, x_2 or x_c

Y is the optimum function value at location X. Stop.

APPENDIX B

THIS IS A SAMPLE OUTPUT OF THE THREE BAR TRUSS PROBLEM AS SOLVED BY THE AUGMENTED LAGRANGE MULTIPLIER METHOD. AN INITIAL PENALTY PARAMETER C=10 AND GAMMA=2 WAS USED. THE PROBLEM SUBPROGRAM IS LISTED FIRST, FOLLOWED BY THE RESULTS.

```
C       3-BAR TRUSS PROBLEM
C
        SUBROUTINE FUNC (ICALC,N,X,FX)
        COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C
        COMMON /CTRL/ CNTR,NDRV,NFUNC,IPRINT
        DIMENSION X(3)
        NCON=13
        N=3
        IF (ICALC.GT.1) GO TO 10
C       INPUT OR INITIALIZATION
        X(1)=1.
        X(2)=1.
        X(3)=1.
        E=1.E7
        RHO=.1
        H=10.
        P=20000.
        WRITE (6,30) X(1),X(2),X(3),E,RHO,H,P
        RETURN
10      CONTINUE
C       EXECUTION
        NFUNC=NFUNC+1
        W=RHO*H*(SQRT(2.)*(X(1)+X(3))+X(2))
        DEN=X(1)*X(2)+X(2)*X(3)+SQRT(2.)*X(1)*X(3)
        SIG11=P/X(1)-X(2)*X(3)*P/( X(1)*DEN)
        SIG21=SQRT(2.)*X(3)*P/DEN
        SIG31=-X(2)*P/DEN
        SIG12=-X(2)*P/DEN
        SIG22=SQRT(2.)*X(1)*P/DEN
        SIG32=P/X(3)-X(1)*X(2)*P/(X(3)*DEN)
        FX=W
        G(1)=SIG11/20000.-1.
        G(2)=SIG21/20000.-1.
        G(3)=SIG31/20000.-1.
        G(4)=SIG12/20000.-1.
        G(5)=SIG22/20000.-1.
        G(6)=SIG32/20000.-1.
        G(7)=-SIG11/15000.-1.
        G(8)=-SIG21/15000.-1.
        G(9)=-SIG31/15000.-1.
```

```
      G(10)=-SIG12/15000.-1.                                                    APP00520
      G(11)=-SIG22/15000.-1.                                                    APP00530
      G(12)=-SIG32/15000.-1.                                                    APP00540
      G(13)=X(1)-X(3)                                                           APP00550
      DO 20 I=1,NCON                                                            APP00560
      ISC(I)=1                                                                  APP00570
20    CONTINUE                                                                  APP00580
      ISC(13)=3                                                                 APP00590
      IF (ICALC.LT.3) RETURN                                                    APP00600
C     OUTPUT                                                                    APP00610
      WRITE (6,40) X(1),X(2),X(3)                                               APP00620
      WRITE (6,50) SIG11,SIG21,SIG31,SIG12,SIG22,SIG32                          APP00630
      WRITE (6,60) W                                                            APP00640
      RETURN                                                                    APP00650
C                                                                               APP00660
30    FORMAT (1H1,15HTHREE BAR TRUSS//5X,5HX(1)=,E12.5/5X,5HX(2)=,E12.5/APP00670
     15X,5HX(3)=,E12.5/5X,2HE=,E12.5/5X,4HRHO=,E12.5/5X,2HH=,E12.5/5X,2HAPP00680
     2P=,E12.5)                                                                 APP00690
40    FORMAT (1H1,4X,5HX(1)=,E12.5/5X,5HX(2)=,E12.5/5X,5HX(3)=,E12.5)    APP00700
50    FORMAT (//5X,6HSIG11=,E12.5/5X,6HSIG21=,E12.5/5X,6HSIG31=,E12.5/5XAPP00710
     1,6HSIG12=,E12.5/5X,6HSIG22=,E12.5/5X,6HSIG32=,E12.5)                      APP00720
60    FORMAT (//5X,7HWEIGHT=,E12.5)                                             APP00730
      END                                                                       APP00740
                                                                                APP00750
                                                                                APP00760
1THREE BAR TRUSS                                                                APP00770
                                                                                APP00780
     X(1)= 0.10000E+01                                                          APP00790
     X(2)= 0.10000E+01                                                          APP00800
     X(3)= 0.10000E+01                                                          APP00810
     E= 0.10000E+08                                                             APP00820
     RHO= 0.10000E+00                                                           APP00830
     H= 0.10000E+02                                                             APP00840
     P= 0.20000E+05                                                             APP00850
0 COUNTER NUMBER    1                                                           APP00860
0    C =  0.10000E+02      CMAX =  0.10000E+07      GAMMA =  0.20000E+01        APP00870
0    X-VECTOR                                                                   APP00880
    0.10000E+01   0.10000E+01   0.10000E+01                                     APP00890
0    OBJ =  0.38284E+01                                                         APP00900
0    CONSTRAINT VALUES                                                          APP00910
  -0.29289E+00  -0.58579E+00  -0.12929E+01  -0.12929E+01  -0.58579E+00         APP00920
  -0.29289E+00  -0.19428E+01  -0.15523E+01  -0.60948E+00  -0.60948E+00         APP00930
  -0.15523E+01  -0.19428E+01   0.0                                             APP00940
0    LAMBDA VECTOR                                                              APP00950
    0.0           0.0           0.0           0.0           0.0                APP00960
   0.0           0.0           0.0           0.0           0.0                 APP00970
   0.0           0.0           0.0                                             APP00980
0   ------------------------------------------------------------------------APP00990
```

```
      UNCONSTRAINED MINIMIZATION RESULTS                                    APP01000
                                                                            APP01010
      FX   0.24984E+01                                                      APP01020
                                                                            APP01030
      X-VECTOR                                                              APP01040
      0.71042E+00    0.37348E+00    0.71042E+00                             APP01050
0     NUMBER OF ITERATIONS=    7                                            APP01060
                                                                            APP01070
      TERMINATION CRITERION                                                 APP01080
           ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS                APP01090
0     ------------------------------------------------------------------    APP01100
0 COUNTER NUMBER    2                                                       APP01110
0     C =  0.20000E+02      CMAX =  0.10000E+07      GAMMA =  0.20000E+01   APP01120
0     X-VECTOR                                                              APP01130
     0.71042E+00   0.37348E+00   0.71042E+00                                APP01140
0     OBJ =  0.24984E+01                                                    APP01150
0     CONSTRAINT VALUES                                                     APP01160
     0.10749E+00  -0.19264E+00  -0.13001E+01  -0.13001E+01  -0.19264E+00   APP01170
    0.10749E+00  -0.24766E+01  -0.20765E+01  -0.59984E+00  -0.59984E+00     APP01180
   -0.20765E+01  -0.24766E+01   0.0                                        APP01190
0     LAMBDA VECTOR                                                         APP01200
     0.10749E+01   0.0           0.0           0.0           0.0           APP01210
    0.10749E+01   0.0           0.0           0.0           0.0            APP01220
    0.0           0.0           0.0                                        APP01230
0     ------------------------------------------------------------------    APP01240
      UNCONSTRAINED MINIMIZATION RESULTS                                    APP01250
                                                                            APP01260
      FX   0.26364E+01                                                      APP01270
                                                                            APP01280
      X-VECTOR                                                              APP01290
      0.78009E+00    0.40891E+00    0.78009E+00                             APP01300
0     NUMBER OF ITERATIONS=    3                                            APP01310
                                                                            APP01320
      TERMINATION CRITERION                                                 APP01330
           ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS                APP01340
           ABS(FX-FY).LT.001 FOR 2 CONSECUTIVE ITERATIONS                   APP01350
0     ------------------------------------------------------------------    APP01360
0 COUNTER NUMBER    3                                                       APP01370
0     C =  0.40000E+02      CMAX =  0.10000E+07      GAMMA =  0.20000E+01   APP01380
0     X-VECTOR                                                              APP01390
     0.78009E+00   0.40891E+00   0.78009E+00                                APP01400
0     OBJ =  0.26364E+01                                                    APP01410
0     CONSTRAINT VALUES                                                     APP01420
     0.90408E-02  -0.26382E+00  -0.12729E+01  -0.12729E+01  -0.26382E+00   APP01430
    0.90408E-02  -0.23454E+01  -0.19816E+01  -0.63618E+00  -0.63618E+00     APP01440
   -0.19816E+01  -0.23454E+01  -0.59605E-07                                APP01450
0     LAMBDA VECTOR                                                         APP01460
     0.12557E+01   0.0           0.0           0.0           0.0           APP01470
```

```
   0.12557E+01   0.0           0.0           0.0           0.0           APP01480
   0.0           0.0          -0.11921E-05                               APP01490
0  ------------------------------------------------------------------------APP01500
     UNCONSTRAINED MINIMIZATION RESULTS                                   APP01510
                                                                          APP01520
     FX   0.26391E+01                                                     APP01530
                                                                          APP01540
     X-VECTOR                                                             APP01550
     0.78191E+00    0.42038E+00    0.78179E+00                            APP01560
0    NUMBER OF ITERATIONS=    2                                          APP01570
                                                                          APP01580
     TERMINATION CRITERION                                                APP01590
          ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS               APP01600
0  ------------------------------------------------------------------------APP01610
0 COUNTER NUMBER    4                                                     APP01620
0    C =  0.80000E+02     CMAX =  0.10000E+07     GAMMA =  0.20000E+01    APP01630
0    X-VECTOR                                                             APP01640
    0.78191E+00   0.42038E+00   0.78179E+00                               APP01650
0    OBJ =  0.26391E+01                                                   APP01660
0    CONSTRAINT VALUES                                                    APP01670
    0.27313E-02  -0.27350E+00  -0.12762E+01  -0.12762E+01  -0.27339E+00   APP01680
   0.28429E-02  -0.23370E+01  -0.19687E+01  -0.63169E+00  -0.63169E+00    APP01690
  -0.19688E+01  -0.23371E+01   0.12040E-03                               APP01700
0    LAMBDA VECTOR                                                        APP01710
    0.13649E+01   0.0           0.0           0.0           0.0          APP01720
   0.13694E+01   0.0           0.0           0.0           0.0           APP01730
   0.0           0.0           0.48149E-02                               APP01740
0  ------------------------------------------------------------------------APP01750
     UNCONSTRAINED MINIMIZATION RESULTS                                   APP01760
                                                                          APP01770
     FX   0.26391E+01                                                     APP01780
                                                                          APP01790
     X-VECTOR                                                             APP01800
     0.78656E+00    0.41182E+00    0.78724E+00                            APP01810
0    NUMBER OF ITERATIONS=    2                                          APP01820
                                                                          APP01830
     TERMINATION CRITERION                                                APP01840
          ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS               APP01850
          ABS(FX-FY).LT.001 FOR 2 CONSECUTIVE ITERATIONS                  APP01860
0  ------------------------------------------------------------------------APP01870
0 COUNTER NUMBER    5                                                     APP01880
0    C =  0.16000E+03     CMAX =  0.10000E+07     GAMMA =  0.20000E+01    APP01890
0    X-VECTOR                                                             APP01900
    0.78656E+00   0.41182E+00   0.78724E+00                               APP01910
0    OBJ =  0.26391E+01                                                   APP01920
0    CONSTRAINT VALUES                                                    APP01930
    0.86594E-03  -0.26939E+00  -0.12703E+01  -0.12703E+01  -0.27002E+00   APP01940
   0.23937E-03  -0.23345E+01  -0.19741E+01  -0.63966E+00  -0.63966E+00    APP01950
```

```
 -0.19733E+01  -0.23337E+01  -0.67466E-03                                    APP01960
0    LAMBDA VECTOR                                                          APP01970
     0.14342E+01    0.0            0.0            0.0            0.0       APP01980
    0.13885E+01    0.0            0.0            0.0            0.0        APP01990
    0.0            0.0           -0.49158E-01                               APP02000
0    ------------------------------------------------------------------     APP02010
     UNCONSTRAINED MINIMIZATION RESULTS                                     APP02020
                                                                            APP02030
     FX   0.26390E+01                                                       APP02040
                                                                            APP02050
     X-VECTOR                                                               APP02060
     0.78835E+00    0.40911E+00    0.78783E+00                              APP02070
0    NUMBER OF ITERATIONS=    4                                             APP02080
                                                                            APP02090
     TERMINATION CRITERION                                                  APP02100
          ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS                 APP02110
          ABS(FX-FY).LT.001 FOR 2 CONSECUTIVE ITERATIONS                    APP02120
0    ------------------------------------------------------------------     APP02130
0 COUNTER NUMBER    6                                                       APP02140
0    C =  0.32000E+03     CMAX =  0.10000E+07     GAMMA =  0.20000E+01      APP02150
0    X-VECTOR                                                               APP02160
    0.78835E+00   0.40911E+00   0.78783E+00                                 APP02170
0    OBJ =  0.26390E+01                                                     APP02180
0    CONSTRAINT VALUES                                                      APP02190
    0.61035E-04  -0.26853E+00  -0.12686E+01  -0.12686E+01  -0.26805E+00     APP02200
   0.54264E-03  -0.23334E+01  -0.19753E+01  -0.64188E+00  -0.64188E+00      APP02210
-0.19759E+01  -0.23341E+01   0.51999E-03                                   APP02220
0    LAMBDA VECTOR                                                          APP02230
     0.14440E+01    0.0            0.0            0.0            0.0       APP02240
    0.14754E+01    0.0            0.0            0.0            0.0        APP02250
    0.0            0.0            0.34040E-01                               APP02260
0    ------------------------------------------------------------------     APP02270
     UNCONSTRAINED MINIMIZATION RESULTS                                     APP02280
                                                                            APP02290
     FX   0.26390E+01                                                       APP02300
                                                                            APP02310
     X-VECTOR                                                               APP02320
     0.78873E+00    0.40778E+00    0.78860E+00                              APP02330
0    NUMBER OF ITERATIONS=    4                                             APP02340
                                                                            APP02350
     TERMINATION CRITERION                                                  APP02360
          ABS(1-FX/FY).LT..001 FOR 2 CONSECUTIVE ITERATIONS                 APP02370
          ABS(FX-FY).LT.001 FOR 2 CONSECUTIVE ITERATIONS                    APP02380
0    ------------------------------------------------------------------     APP02390
1FINAL OPTIMIZATION INFORMATION                                             APP02400
                                                                            APP02410
     FX =  0.26390E+01                                                      APP02420
                                                                            APP02430
```

```
      X-VECTOR                                                                  APP02440
     0.78873E+00   0.40778E+00   0.78860E+00                                    APP02450
0     NUMBER OF ITERATIONS=      6                                             APP02460
      FUNCTION EVALUATED  185 TIMES                                            APP02470
                                                                               APP02480
      TERMINATION CRITERION                                                    APP02490
0          ABS(FX-FY).LT..001 FOR 3 CONSECUTIVE ITERATIONS                     APP02500
0          ABS(1.-FX/FY).LT..001 FOR 3 CONSECUTIVE ITERATIONS                  APP02510
1     X(1)= 0.78873E+00                                                        APP02520
      X(2)= 0.40778E+00                                                        APP02530
      X(3)= 0.78860E+00                                                        APP02540
                                                                               APP02550
                                                                               APP02560
      SIG11= 0.20002E+05                                                       APP02570
      SIG21= 0.14647E+05                                                       APP02580
      SIG31=-0.53556E+04                                                       APP02590
      SIG12=-0.53556E+04                                                       APP02600
      SIG22= 0.14649E+05                                                       APP02610
      SIG32= 0.20005E+05                                                       APP02620
                                                                               APP02630
                                                                               APP02640
      WEIGHT= 0.26385E+01                                                      APP02650
```

APPENDIX C

```
THIS IS A LISTING OF THE AUGMENTED LAGRANGE MULTIPLIER PROGRAM AND ITS
ASSOCIATED SUBPROGRAMS.  THE PROGRAM REQUIRES AN ADDITIONAL USER
SUPPLIED SUBPROGRAM TO INPUT THE DESIRED PROBLEM.  THE PROGRAM MUST BE
OF THE FOLLOWING FORM:

      SUBROUTINE FCN (ICALC,N,X,FX)
      COMMON/LAGR/ NCON,G(40),ISC(40),XLAM(40),C
      COMMON/CTRL/ CNTR,NDRV,NFUNC,IPRINT
      IF (ICALC.GT.1)GO TO 10OBJECTIVE FUNCTION.  THE SECOND CARD IS,
C     INPUT OR INITIALIZATION OF DATA
10    CONTINUE
C     EXECUTION
      (USER SUPPLIED OBJECTIVE AND NORMALIZED CONSTRAINT FUNCTIONS)
      (DEFINE CONSTRAINT TYPE SETTING ISC(I)= 1 FOR EQUALITY CONSTRAINT
      OF ISC(I)= 3 FOR INEQUALITY CONSTRAINT, WHERE I IS THE CONSTRAINT
      NUMBER)
      NFUNC= NFUNC+1
      IF (ICALC.LT.3) RETURN
C     OUTPUT
      RETURN
      END

THE FOLLOWING IS THE AUGMENTED LAGRANGE MULTIPLIER PROGRAM.

C     MAIN PROGRAM FOR EXECUTION OF LAGRANGE MULTIPLIER METHOD
C
      DIMENSION X(10),Y(10),S(10),H(10,10),GX(10),GY(10),DELG(10),DELX(1
     10)
      COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C
      EXTERNAL LAGR1
      NDV=10
      NH=10
      CALL LGRANG (NDV,X,FX,Y,S,H,NH,GX,GY,DELG,DELX)
      STOP
      END
      SUBROUTINE LGRANG (NDV,X,FX,Y,S,H,NH,GX,GY,DELG,DELX)
C     ***************************************************************
C     FUNCTION MINIMIZATION BY AUGMENTED LAGRANGE MULTIPLIER METHOD
C     ***************************************************************
C     BY L. E. MADSEN                                    MARCH, 1981
C     NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.
C     ***************************************************************
C     PRINT CONTROL
C
C     0-  INPUT/OUTPUT ONLY
```

```
C       1-  LAGRANGE MULTIPLIER INFORMATION PLUS INPUT/OUTPUT                  APP00520
C       2-  UNCONSTRAINED MINIMIZATION, LAGRANGE MULTIPLIER, INPUT/OUTPUT      APP00530
C       3-  ONE DIMENSIONAL SEARCH,UNCONSTRAINED MINIMIZATION, LAGRANGE        APP00540
C           MULTIPLIER,INPUT/OUTPUT INFORMATION                                APP00550
C       8-  DEBUG UNCONSTRAINED MINIMIZATION PLUS ABOVE                        APP00560
C       9-  DEBUG ONE DIMENSIONAL SEARCH PLUS ABOVE                            APP00570
C       ***************************************************************      APP00580
        DIMENSION X(1),Y(1),S(1),H(NH,1),GX(1),GY(1),DELG(1),DELX(1)           APP00590
        COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                   APP00600
        COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C                            APP00610
        EXTERNAL LAGR1                                                         APP00620
        NCTR=0                                                                 APP00630
        NDRV=0                                                                 APP00640
        NFUNC=0                                                                APP00650
        IPRINT=9                                                               APP00660
        ITMAX=2                                                                APP00670
        KOBJ1=0                                                                APP00680
        KOBJ2=0                                                                APP00690
        KOBJ3=0                                                                APP00700
        ITRM=3                                                                 APP00710
        LFLG=0                                                                 APP00720
        DO 10 I=1,2                                                            APP00730
10      CALL FUNC (I,NDV,X,FX)                                                 APP00740
        NH=NDV                                                                 APP00750
        CMAX=1.E6                                                              APP00760
        C=1.E2                                                                 APP00770
        GAMMA=1.                                                               APP00780
        DO 20 I=1,NCON                                                         APP00790
        XLAM(I)=0.                                                             APP00800
20      CONTINUE                                                               APP00810
        NCTR=NCTR+1                                                            APP00820
        IF (IPRINT.GE.1) WRITE (6,110) NCTR                                    APP00830
        IF (IPRINT.GE.1) WRITE (6,120) C,CMAX,GAMMA                            APP00840
        IF (IPRINT.GE.1) WRITE (6,130)                                         APP00850
        IF (IPRINT.GE.1) WRITE (6,140) (X(I),I=1,NDV)                          APP00860
        IF (IPRINT.GE.1) WRITE (6,150) FX                                      APP00870
        IF (IPRINT.GE.1) WRITE (6,160)                                         APP00880
        IF (IPRINT.GE.1) WRITE (6,140) (G(I),I=1,NCON)                         APP00890
        IF (IPRINT.GE.1) WRITE (6,170)                                         APP00900
        IF (IPRINT.GE.1) WRITE (6,140) (XLAM(I),I=1,NCON)                      APP00910
        CALL DFPEX (NDV,X,Y,S,H,NH,GX,GY,DELG,DELX,FY,LAGR1)                   APP00920
C                                                                              APP00930
C       CHECK CONVERGENCE BY FUNCTION DIFFERENCE                               APP00940
        IF (NCTR.LT.ITMAX) GO TO 30                                            APP00950
        KOBJ3=1                                                                APP00960
        GO TO 100                                                              APP00970
30      KOBJ1=KOBJ1+1                                                          APP00980
        IF (ABS(FX-FY).GE..001) KOBJ1=0                                        APP00990
```

```
      KOBJ2=KOBJ2+1                                                          APP01000
      IF (ABS(1.-FX/(FY+1.E-10)).GE..001) KOBJ2=0                            APP01010
      IF (KOBJ1.GE.ITRM.OR.KOBJ2.GE.ITRM) GO TO 100                          APP01020
      DO 40 I=1,NDV                                                          APP01030
      X(I)=Y(I)                                                              APP01040
40    CONTINUE                                                               APP01050
      FX=FY                                                                  APP01060
C                                                                            APP01070
C     UPDATE LAMBDA VECTOR                                                   APP01080
      IF (LFLG.GE.1) GO TO 90                                                APP01090
      DO 80 I=1,NCON                                                         APP01100
      II=ISC(I)                                                              APP01110
      IF (II.GT.2) GO TO 50                                                  APP01120
C      INEQUALITY CONSTRAINT                                                 APP01130
      PSI=-XLAM(I)/C                                                         APP01140
      IF (G(I).GT.PSI) PSI=G(I)                                              APP01150
      XLAM(I)=XLAM(I)+C*PSI                                                  APP01160
      GO TO 70                                                               APP01170
50    IF (II.GT.4) GO TO 60                                                  APP01180
C      EQUALITY CONSTRAINT                                                   APP01190
      XLAM(I)=XLAM(I)+C*G(I)                                                 APP01200
      GO TO 70                                                               APP01210
60    WRITE (6,180) I,ISC(I)                                                 APP01220
70    IF (ABS(XLAM(I)).LT.1.E-15) XLAM(I)=0.                                 APP01230
      IF (XLAM(I).GE.1.E15) XLAM(I)=1.E15                                    APP01240
      IF (XLAM(I).LT.-1.E15) XLAM(I)=-1.E15                                  APP01250
80    CONTINUE                                                               APP01260
90    C=GAMMA*C                                                              APP01270
      IF (C.GT.CMAX) C=CMAX                                                  APP01280
      GO TO 20                                                               APP01290
100   WRITE (6,190) FY                                                       APP01300
      WRITE (6,140) (Y(I),I=1,NDV)                                           APP01310
      WRITE (6,200) NCTR,NFUNC                                               APP01320
      IF (KOBJ3.GT.0) WRITE (6,210)                                          APP01330
      IF (KOBJ1.GT.0) WRITE (6,220) ITRM                                     APP01340
      IF (KOBJ2.GT.0) WRITE (6,230) ITRM                                     APP01350
      ICALC=3                                                                APP01360
      CALL FUNC (ICALC,NDV,Y,FY)                                             APP01370
      RETURN                                                                 APP01380
C                                                                            APP01390
110   FORMAT (1HO,15H COUNTER NUMBER,I5)                                     APP01400
120   FORMAT (1HO,4X,4HC = ,E12.5,5X,7HCMAX = ,E12.5,5X,8HGAMMA = ,E12.5APP01410
     1)                                                                      APP01420
130   FORMAT (1HO,4X,8HX-VECTOR)                                             APP01430
140   FORMAT (1H ,5(5X,E12.5))                                               APP01440
150   FORMAT (1HO,4X,6HOBJ = ,E12.5)                                         APP01450
160   FORMAT (1HO,4X,17HCONSTRAINT VALUES)                                   APP01460
170   FORMAT (1HO,4X,13HLAMBDA VECTOR)                                       APP01470
```

```
180   FORMAT (1H0,23HERROR:  ISC.GT.4;  ISC(,I3,3H)= ,I3)                    APP01480
190   FORMAT (1H1,30HFINAL OPTIMIZATION INFORMATION//5X,5HFX = ,E12.5//5APP01490
     1X,8HX-VECTOR)                                                          APP01500
200   FORMAT (1H0,4X,22HNUMBER OF ITERATIONS= ,I5/5X,18HFUNCTION EVALUATAPP01510
     1ED,I5,6H TIMES//5X,21HTERMINATION CRITERION)                           APP01520
210   FORMAT (1H0,9X,37HMAXIMUM NUMBER OF ITERATIONS EXCEEDED)                APP01530
220   FORMAT (1H0,9X,22HABS(FX-FY).LT..001 FOR,I2,23H CONSECUTIVE ITERATAPP01540
     1IONS)                                                                  APP01550
230   FORMAT (1H0,9X,25HABS(1.-FX/FY).LT..001 FOR,I2,23H CONSECUTIVE ITEAPP01560
     1RATIONS)                                                               APP01570
      END                                                                    APP01580
      SUBROUTINE LAGR1 (NDV,X,OBJ)                                           APP01590
      DIMENSION X(10)                                                        APP01600
      COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C                            APP01610
      COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                   APP01620
      ICALC=2                                                                APP01630
      CALL FUNC (ICALC,NDV,X,OBJ)                                            APP01640
      IF (IPRINT.GE.9) WRITE (6,40) OBJ                                      APP01650
      IF (IPRINT.GE.9) WRITE (6,50) (G(I),I=1,NCON)                          APP01660
      DO 30 I=1,NCON                                                         APP01670
      II=ISC(I)                                                              APP01680
      IF (II.GT.2) GO TO 10                                                  APP01690
C     INEQUALITY CONSTRAINT                                                  APP01700
      PSI=-XLAM(I)/C                                                         APP01710
      IF (G(I).GT.PSI) PSI=G(I)                                              APP01720
      OBJ=OBJ+XLAM(I)*PSI+.5*C*PSI**2                                        APP01730
      GO TO 30                                                               APP01740
10    IF (II.GT.4) GO TO 20                                                  APP01750
C     EQUALITY CONSTRAINT                                                    APP01760
      OBJ=OBJ+XLAM(I)*G(I)+.5*C*G(I)**2                                      APP01770
      GO TO 30                                                               APP01780
C     CONSTRAINT IS INDETERMINATE                                            APP01790
20    WRITE (6,60) ISC(I)                                                    APP01800
      IF (IPRINT.GE.9) WRITE (6,70) OBJ                                      APP01810
30    CONTINUE                                                               APP01820
      RETURN                                                                 APP01830
C                                                                            APP01840
40    FORMAT (1H0,4X,4HFX= ,E12.5//5X,17HCONSTRAINT VALUES)                  APP01850
50    FORMAT (1H ,5(3X,E12.5))                                               APP01860
60    FORMAT (1H0,34HERROR:  ISC UNDEFINED.  ISC VALUES/5(3X,E12.5))         APP01870
70    FORMAT (1H0,4X,5HOBJ= ,E12.5)                                          APP01880
      END                                                                    APP01890
      SUBROUTINE DFPEX (N,X,Y,S,H,NH,GX,GY,DELG,DELX,FY,FCN)                 APP01900
C     ***************************************************************APP01910
C     ROUTINE FOR DAVIDON-FLETCHER-POWELL METHOD FOR UNCONSTRAINED           APP01920
C     MINIMIZATION                                                           APP01930
C     ***************************************************************APP01940
C     BY L. E. MADSEN                                  MARCH,1981            APP01950
```

```
C     NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                          APP01960
C     ***********************************************************************APP01970
C     DEFINITION OF PARAMETERS:                                            APP01980
C        N       -NUMBER OF DESIGN VARIABLES                               APP01990
C        X       -DESIGN VARIABLE ARRAY                                    APP02000
C        S       -SEARCH DIRECTION VECTOR                                  APP02010
C        H       -HESSIAN MATRIX APPROXIMATION                             APP02020
C        GX      -GRADIENT VECTOR                                          APP02030
C        DELG    -GRADIENT DIFFERENCE VECTOR                               APP02040
C        DELX    -VARIABLE DIFFERENCE VECTOR                               APP02050
C        FX      -OBJECTIVE FUNCTION                                       APP02060
C        ITER    -ITERATION COUNTER                                        APP02070
C        NFUNC   -FUNCTION EVALUATION COUNTER                              APP02080
C        NDRV    -DERIVATIVE EVALUATION COUNTER                            APP02090
C        IPRINT  -PRINT CONTROL                                            APP02100
C     ***********************************************************************APP02110
      EXTERNAL FCN                                                         APP02120
      COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                 APP02130
      COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C                          APP02140
      DIMENSION X(10),Y(10),S(10),H(NH,10),GY(10),GX(10),DELG(10),DELX(1APP02150
     10)                                                                   APP02160
C                                                                          APP02170
C     INITIALIZE CONTROL PARAMETERS                                        APP02180
      ITER=0                                                               APP02190
      KOBJ1=0                                                              APP02200
      KOBJ2=0                                                              APP02210
      KOBJ3=0                                                              APP02220
      ITMAX=30                                                             APP02230
      SCAL1=0.                                                             APP02240
      ITRM=2                                                               APP02250
      KOUNT=0                                                              APP02260
      DX=.01                                                               APP02270
C                                                                          APP02280
C     EVALUATE INITIAL FUNCTION AND DERIVATIVE                             APP02290
      CALL FCN (N,X,FX)                                                    APP02300
      CALL DERIV (N,X,DX,FX,GX,FCN)                                        APP02310
      ICNDIR=N+1                                                           APP02320
      IF (IPRINT.GE.2) WRITE (6,140)                                       APP02330
C                                                                          APP02340
C     SET UP IDENTITY MATRIX                                               APP02350
10    NEWH=0                                                               APP02360
      IFLAG=1                                                              APP02370
      DO 30 I=1,N                                                          APP02380
      DO 20 J=1,N                                                          APP02390
20    H(I,J)=0.                                                            APP02400
30    H(I,I)=1.                                                            APP02410
C                                                                          APP02420
C     TAKE GRADIENT STEP                                                   APP02430
```

```
      DO 40 I=1,N                                                       APP02440
      S(I)=-GX(I)                                                       APP02450
      SMAX=ABS(S(I))                                                    APP02460
      IF (SMAX.GT.SCAL1) SCAL1=SMAX                                     APP02470
   40 CONTINUE                                                          APP02480
      NEWH=NEWH+1                                                       APP02490
      IF (NEWH.GE.ICNDIR) GO TO 10                                      APP02500
      ITER=ITER+1                                                       APP02510
      IF (IPRINT.GE.2) WRITE (6,150) ITER                               APP02520
      IF (IPRINT.GE.2) WRITE (6,160)                                    APP02530
      IF (IPRINT.GE.2) WRITE (6,170) (GX(I),I=1,N)                      APP02540
      IF (IPRINT.GE.8) WRITE (6,180)                                    APP02550
      IF (IPRINT.GE.8) WRITE (6,170) (S(I),I=1,N)                       APP02560
      IF (IPRINT.GE.8) WRITE (6,190)                                    APP02570
      DO 50 I=1,N                                                       APP02580
      IF (IPRINT.GE.8) WRITE (6,170) (H(I,J),J=1,N)                     APP02590
   50 CONTINUE                                                          APP02600
      SDF=0.                                                            APP02610
      DO 60 I=1,N                                                       APP02620
      SDF=SDF+S(I)*GX(I)                                                APP02630
   60 CONTINUE                                                          APP02640
      IF (IPRINT.GE.8) WRITE (6,200) SDF                                APP02650
      IF (SDF.GE.0.) GO TO 10                                           APP02660
      SCAL=SCAL1                                                        APP02670
      SCAL1=0.                                                          APP02680
      IF (IPRINT.GE.8) WRITE (6,210) SCAL                               APP02690
      DO 70 I=1,N                                                       APP02700
      S(I)=S(I)/SCAL                                                    APP02710
   70 CONTINUE                                                          APP02720
C                                                                       APP02730
C     FIND NEXT POINT, X                                                APP02740
      CALL GOLDS (N,X,FX,S,Y,FY,FCN)                                    APP02750
      IF (IPRINT.GE.2) WRITE (6,220) FY                                 APP02760
      IF (IPRINT.GE.2) WRITE (6,170) (Y(I),I=1,N)                       APP02770
      IF (FX.GE.FY) GO TO 80                                            APP02780
      KOUNT=KOUNT+1                                                     APP02790
      IF (KOUNT.GT.2) GO TO 80                                          APP02800
      IF (IPRINT.GE.8) WRITE (6,230)                                    APP02810
      GO TO 10                                                          APP02820
   80 CONTINUE                                                          APP02830
C                                                                       APP02840
C     CHECK CONVERGENCE                                                 APP02850
      IF (ITER.LT.ITMAX) GO TO 90                                       APP02860
      KOBJ3=1                                                           APP02870
      GO TO 130                                                         APP02880
   90 KOBJ1=KOBJ1+1                                                     APP02890
      IF (ABS(1.-FX/(FY-1.E-9)).GE..001) KOBJ1=0                        APP02900
      KOBJ2=KOBJ2+1                                                     APP02910
```

```
      IF (ABS(FX-FY).GE..001) KOBJ2=0                                    APP02920
      IF (KOBJ1.GE.ITRM.OR.KOBJ2.GE.ITRM) GO TO 130                      APP02930
C                                                                        APP02940
C     CONVERGENCE CRITERIA NOT SATISFIED.  FIND A NEW DIRECTION MATRIX   APP02950
      CALL DERIV (N,Y,DX,FY,GY,FCN)                                      APP02960
      DO 100 I=1,N                                                       APP02970
      DELG(I)=GY(I)-GX(I)                                                APP02980
      DELX(I)=Y(I)-X(I)                                                  APP02990
      GX(I)=GY(I)                                                        APP03000
100   X(I)=Y(I)                                                          APP03010
      FX=FY                                                              APP03020
      IF ((KOBJ1.EQ.0.OR.KOBJ2.EQ.0).AND.IFLAG.NE.1) GO TO 10            APP03030
C                                                                        APP03040
C     CHECK FY FOR MINIMUM IN S-DIRECTION.  SKIP H UPDATE, IF NOT.       APP03050
      SDF2=0.                                                            APP03060
      DO 110 I=1,N                                                       APP03070
      SDF2=SDF2+S(I)*GY(I)                                               APP03080
110   CONTINUE                                                           APP03090
      IF (IPRINT.GE.8) WRITE (6,240) SDF2                                APP03100
      IFLAG=IFLAG+1                                                      APP03110
      IF (ABS(SDF2).LT..01.OR.IFLAG.GE.2) IFLAG=0                        APP03120
      IF (IFLAG.EQ.0) CALL DFP01 (N,H,NH,DELX,DELG)                      APP03130
C                                                                        APP03140
C     SET UP NEW SEARCH DIRECTION                                        APP03150
      DO 120 I=1,N                                                       APP03160
      S(I)=0.                                                            APP03170
      DO 120 J=1,N                                                       APP03180
      S(I)=S(I)-H(I,J)*GY(J)                                             APP03190
      SMAX=ABS(S(I))                                                     APP03200
      IF (SMAX.GT.SCAL1) SCAL1=SMAX                                      APP03210
120   CONTINUE                                                           APP03220
      GO TO 40                                                           APP03230
C                                                                        APP03240
C     CONVERGENCE TEST SATISFIED                                         APP03250
130   IF (IPRINT.GE.1) WRITE (6,250)                                     APP03260
      IF (IPRINT.GE.1) WRITE (6,260) FY                                  APP03270
      IF (IPRINT.GE.1) WRITE (6,170) (Y(I),I=1,N)                        APP03280
      IF (IPRINT.GE.1) WRITE (6,270) ITER                                APP03290
      IF (IPRINT.GE.1.AND.KOBJ3.GT.0) WRITE (6,280)                      APP03300
      IF (IPRINT.GE.1.AND.KOBJ1.GT.0) WRITE (6,290) ITRM                 APP03310
      IF (IPRINT.GE.1.AND.KOBJ2.GT.0) WRITE (6,300) ITRM                 APP03320
      IF (IPRINT.GE.1) WRITE (6,250)                                     APP03330
      RETURN                                                             APP03340
C                                                                        APP03350
140   FORMAT (1H0,39HUNCONSTRAINED MINIMIZATION:  DFP METHOD)            APP03360
150   FORMAT (1H0,10HITERATION=,I5)                                      APP03370
160   FORMAT (1H0,4X,15HGRADIENT VECTOR)                                 APP03380
170   FORMAT (1H ,5(3X,E12.5))                                           APP03390
```

```
180    FORMAT (1HO,4X,8HS-VECTOR)                                                APP03400
190    FORMAT (1HO,4X,8HH-MATRIX)                                                APP03410
200    FORMAT (1HO,4X,5HSDF   ,E12.5/)                                           APP03420
210    FORMAT (1H ,4X,6HSCAL  ,E12.5)                                            APP03430
220    FORMAT (1HO,4X,4HFX    ,E12.5//5X,8HX-VECTOR)                             APP03440
230    FORMAT (1HO,4X,19HFY.GT.FX.  RESET H.)                                    APP03450
240    FORMAT (1HO,4X,6HSDF2  ,E12.5)                                            APP03460
250    FORMAT (1HO,4X,68(1H-))                                                   APP03470
260    FORMAT (1H ,4X,34HUNCONSTRAINED MINIMIZATION RESULTS//5X,4HFX  ,E1APP03480
      12.5//5X,8HX-VECTOR)                                                       APP03490
270    FORMAT (1HO,4X,21HNUMBER OF ITERATIONS=,I5//5X,21HTERMINATION CRITAPP03500
      1ERION)                                                                    APP03510
280    FORMAT (1H ,9X,37HMAXIMUM NUMBER OF ITERATIONS EXCEEDED)                  APP03520
290    FORMAT (1H ,9X,24HABS(1-FX/FY).LT..001 FOR,I2,23H CONSECUTIVE ITERAPP03530
      1ATIONS)                                                                   APP03540
300    FORMAT (1H ,9X,21HABS(FX-FY).LT.001 FOR,I2,23H CONSECUTIVE ITERATIAPP03550
      1ONS)                                                                      APP03560
       END                                                                       APP03570
       SUBROUTINE DFP01 (N,H,NH,DELX,DELG)                                       APP03580
C      ***************************************************************APP03590
C      ROUTINE TO APPROXIMATE THE INVERSE HESSIAN MATRIX BY DAVIDON-             APP03600
C      FLETCHER-POWELL METHOD                                                    APP03610
C      ***************************************************************APP03620
C      BY L. E. MADSEN                                    MARCH,1981             APP03630
C      NAVAL POSTGRADUATE SCHOOL,MONTEREY, CALIF.                                APP03640
C      ***************************************************************APP03650
       COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                      APP03660
       DIMENSION HDG(10),DGH(10),DELX(1),DELG(1),H(NH,1),A(10,10),B(10,10APP03670
      1)                                                                         APP03680
       DO 10 I=1,N                                                               APP03690
       DO 10 J=1,N                                                               APP03700
       A(I,J)=0.                                                                 APP03710
       B(I,J)=0.                                                                 APP03720
10     CONTINUE                                                                  APP03730
       DXDG=1.E-9                                                                APP03740
       DGHDG=1.E-9                                                               APP03750
       DO 30 I=1,N                                                               APP03760
       HDG(I)=0.                                                                 APP03770
       DGH(I)=0.                                                                 APP03780
       DO 20 J=1,N                                                               APP03790
       HDG(I)=HDG(I)-H(I,J)*DELG(J)                                              APP03800
20     DGH(I)=DGH(I)+DELG(J)*H(J,I)                                              APP03810
       DXDG=DXDG+DELX(I)*DELG(I)                                                 APP03820
30     DGHDG=DGHDG+DGH(I)*DELG(I)                                                APP03830
       DO 40 I=1,N                                                               APP03840
       DO 40 J=1,N                                                               APP03850
       A(I,J)=A(I,J)+DELX(I)*DELX(J)/DXDG                                        APP03860
       B(I,J)=B(I,J)+HDG(I)*DGH(J)/DGHDG                                         APP03870
```

```
40    H(I,J)=H(I,J)+DELX(I)*DELX(J)/DXDG+HDG(I)*DGH(J)/DGHDG                  APP03880
      IF (IPRINT.GE.8) WRITE (6,70)                                          APP03890
      DO 50 I=1,N                                                            APP03900
      IF (IPRINT.GE.8) WRITE (6,80) (A(I,J),J=1,N)                           APP03910
50    CONTINUE                                                               APP03920
      IF (IPRINT.GE.8) WRITE (6,90)                                          APP03930
      DO 60 I=1,N                                                            APP03940
      IF (IPRINT.GE.8) WRITE (6,80) (B(I,J),J=1,N)                           APP03950
60    CONTINUE                                                               APP03960
      RETURN                                                                 APP03970
C                                                                            APP03980
70    FORMAT (1HO,4X,8HA-VECTOR)                                             APP03990
80    FORMAT (1H ,5(3X,E12.5))                                               APP04000
90    FORMAT (1HO,4X,8HB-VECTOR)                                             APP04010
      END                                                                    APP04020
      SUBROUTINE DERIV (N,X,DX,FX,GX,FCN)                                    APP04030
C     ***************************************************************      APP04040
CC    ROUTINE TO CALCULATE THE DERIVATIVE BY FINITE DIFFERENCE METHOD        APP04050
CC    ***************************************************************      APP04060
CC    BY L. E. MADSEN                                  MARCH,1981            APP04070
CC    NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                            APP04080
C     ***************************************************************      APP04090
      EXTERNAL FCN                                                           APP04100
      COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                   APP04110
      COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C                            APP04120
      DIMENSION X(1),GX(1)                                                   APP04130
      DO 10 I=1,N                                                            APP04140
      XI=X(I)                                                                APP04150
      X(I)=X(I)+DX                                                           APP04160
      CALL FCN (N,X,FXI)                                                     APP04170
      GX(I)=(FXI-FX)/DX                                                      APP04180
      X(I)=XI                                                                APP04190
10    CONTINUE                                                               APP04200
      NDRV=NDRV+N                                                            APP04210
      RETURN                                                                 APP04220
      END                                                                    APP04230
      SUBROUTINE GOLDS (N,XOLD,YOLD,S,XNEW,YNEW,FCN)                         APP04240
C     ***************************************************************      APP04250
CC    SOLVES ONE DIMENSIONAL SEARCH FOR UNCONSTRAINED MINIMIZATION BY        APP04260
CC    GOLDEN SECTION METHOD                                                  APP04270
CC    ***************************************************************      APP04280
CC    BY L. E. MADSEN                                  MARCH, 1981           APP04290
CC    NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                            APP04300
C     ***************************************************************      APP04310
      EXTERNAL FCN                                                           APP04320
      COMMON /CTRL/ NCTR,NDRV,NFUNC,IPRINT                                   APP04330
      COMMON /LAGR/ NCON,G(40),ISC(40),XLAM(40),C                            APP04340
      DIMENSION S(1),XOLD(1),XNEW(1),FX(20),ALPHA(20)                        APP04350
```

```
      NGLD=1                                                            APP04360
      TAU=.38196601                                                     APP04370
      EPTOL=.06                                                         APP04380
C                                                                       APP04390
C     FIND BRACKETS ON MINIMUM OF 1-D FUNCTION                          APP04400
      XMULT=1.61803399                                                  APP04410
      XADD=.1                                                           APP04420
      ALPHA(1)=.0                                                       APP04430
      FX(1)=YOLD                                                        APP04440
10    IF (NGLD.GT.1) XADD=0.                                            APP04450
      CALL ADS006 (ALPHA,FX,NGLD,XMULT,XADD,ALPHNU)                     APP04460
      IF (ALPHNU.LT.0) GO TO 20                                         APP04470
      CALL ADS010 (N,ALPHNU,XOLD,S,XNEW)                                APP04480
      CALL FCN (N,XNEW,YNEW)                                            APP04490
      IF (IPRINT.GE.9) WRITE (6,120) NGLD,ALPHNU,YNEW                   APP04500
      NGLD=NGLD+1                                                       APP04510
      ALPHA(NGLD)=ALPHNU                                                APP04520
      FX(NGLD)=YNEW                                                     APP04530
      GO TO 10                                                          APP04540
20    ALPHU=ALPHA(NGLD)                                                 APP04550
      IF (NGLD.GT.2) GO TO 30                                           APP04560
      ALPHL=ALPHA(1)                                                    APP04570
      YL=YOLD                                                           APP04580
      YU=YNEW                                                           APP04590
      GO TO 40                                                          APP04600
30    ALPHL=ALPHA(NGLD-2)                                               APP04610
      YL=FX(NGLD-2)                                                     APP04620
      Y1=FX(NGLD-1)                                                     APP04630
      YU=FX(NGLD)                                                       APP04640
40    CONTINUE                                                          APP04650
C                                                                       APP04660
C     DETERMINE CONVERGENCE TOLERANCE                                   APP04670
      EPS=EPTOL/(ALPHU-ALPHL)                                           APP04680
C                                                                       APP04690
C     CALCULATE NUMBER OF FUNCTION EVALUATIONS REQUIRED TO ACHIEVE      APP04700
C     CONVERGENCE TOLERANCE                                             APP04710
      XN=ALOG(EPS)/ALOG(1.-TAU)+3.                                      APP04720
      NEVAL=IFIX(XN)+1                                                  APP04730
      IF (IPRINT.GE.3) WRITE (6,130) NEVAL                              APP04740
C                                                                       APP04750
C     CALCULATE INTERIOR POINTS, ALPH1 AND ALPH2                        APP04760
      CALL ADS007 (ALPHL,ALPH1,ALPH2,ALPHU)                             APP04770
C                                                                       APP04780
C     EVALUATE FUNCTION AT INTERIOR POINTS, X1 AND X2                   APP04790
      CALL ADS010 (N,ALPH1,XOLD,S,XNEW)                                 APP04800
      IF (NGLD.LE.2) CALL FCN (N,XNEW,Y1)                               APP04810
      CALL ADS010 (N,ALPH2,XOLD,S,XNEW)                                 APP04820
      CALL FCN (N,XNEW,Y2)                                              APP04830
```

```
      IF (IPRINT.GE.9) WRITE (6,140) ALPHL,YL,ALPH1,Y1,ALPH2,Y2,ALPHU,YUAPP04840
C                                                                       APP04850
C     INITIALIZE COUNTER                                                APP04860
      K=4                                                               APP04870
      IF (NEVAL.LE.4) GO TO 80                                          APP04880
C                                                                       APP04890
C     UPDATE BOUNDS ON THE MINIMUM AND ESTIMATE A NEW INTERIOR POINT    APP04900
50    CALL ADS008 (ALPHL,YL,ALPH1,Y1,ALPH2,Y2,ALPHU,YU,IX)              APP04910
      IF (IX.EQ.1) GO TO 60                                             APP04920
      CALL ADS010 (N,ALPH2,XOLD,S,XNEW)                                 APP04930
      CALL FCN (N,XNEW,Y2)                                              APP04940
      GO TO 70                                                          APP04950
60    CALL ADS010 (N,ALPH1,XOLD,S,XNEW)                                 APP04960
      CALL FCN (N,XNEW,Y1)                                              APP04970
70    CONTINUE                                                          APP04980
      K=K+1                                                             APP04990
      IF (IPRINT.GE.9) WRITE (6,150) K                                  APP05000
      IF (IPRINT.GE.9) WRITE (6,140) ALPHL,YL,ALPH1,Y1,ALPH2,Y2,ALPHU,YUAPP05010
C                                                                       APP05020
C     CHECK CONVERGENCE                                                 APP05030
      IF (K.LT.NEVAL) GO TO 50                                          APP05040
C                                                                       APP05050
C     COMPARE GOLDEN SECTION TO CUBIC INTERPOLATION                     APP05060
      ZRO= 1.E-5                                                        APP05070
      CALL ADS003 (2,ALPHL,YL,YL,ALPH1,Y1,ALPH2,Y2,ALPHU,YU,ZRO,AZ,A1,A2APP05080
     1,A3)                                                              APP05090
      IF (IPRINT.GE.9) WRITE (6,160) AZ,A1,A2,A3                        APP05100
      IF (A2.EQ.0.) GO TO 80                                            APP05110
      CALL ADS005 (3,A1,A2,A3,ALPHMN,ALPHMX)                            APP05120
      IF (ALPHMN.EQ.-1.E+20) GO TO 80                                   APP05130
      CALL ADS010 (N,ALPHMN,XOLD,S,XNEW)                                APP05140
      CALL FCN (N,XNEW,YPOLY)                                           APP05150
      IF (YPOLY.GT.YL.AND.YPOLY.GT.Y1.AND.YPOLY.GT.Y2) GO TO 80         APP05160
      YNEW=YPOLY                                                        APP05170
      IF (IPRINT.GE.3) WRITE (6,170)                                    APP05180
      GO TO 110                                                         APP05190
80    IF (YL.GT.Y1.AND.YL.GT.Y2) GO TO 90                               APP05200
      YNEW=YL                                                           APP05210
      CALL ADS010 (N,ALPHL,XOLD,S,XNEW)                                 APP05220
      GO TO 110                                                         APP05230
90    IF (Y1.GT.Y2) GO TO 100                                           APP05240
      YNEW=Y1                                                           APP05250
      CALL ADS010 (N,ALPH1,XOLD,S,XNEW)                                 APP05260
      GO TO 110                                                         APP05270
100   YNEW=Y2                                                           APP05280
      CALL ADS010 (N,ALPH2,XOLD,S,XNEW)                                 APP05290
110   IF (IPRINT.GE.3) WRITE (6,140) ALPHL,YL,ALPH1,Y1,ALPH2,Y2,ALPHU,YUAPP05300
      IF (IPRINT.GE.3) WRITE (6,180) K                                  APP05310
```

```
      RETURN                                                                APP05320
C                                                                           APP05330
120   FORMAT (1HO,4X,5HNGLD=,I5/5X,7HALPHNU=,E12.5/5X,5HYNEW=,E12.5)        APP05340
130   FORMAT (1HO,4X,23HGOLDEN SECTION SEARCH. ,I5,22H EVALUATIONS REQUI    APP05350
     1RED.)                                                                 APP05360
140   FORMAT (1HO,4X,6HALPHL=,E12.5/5X,6HYL    =,E12.5/5X,6HALPH1=,E12.5/   APP05370
     15X,6HY1=    ,E12.5/5X,6HALPH2=,E12.5/5X,6HY2=    ,E12.5/5X,6HALPHU=,   APP05380
     2E12.5/5X,6HYU=    ,E12.5)                                             APP05390
150   FORMAT (1HO,4X,3HK= ,I5)                                              APP05400
160   FORMAT (1HO,4X,17HCUBIC COEFICIENTS/5X,4(E12.5,5X))                   APP05410
170   FORMAT (1HO,4X,24HCUBIC APPROXIMATION USED)                           APP05420
180   FORMAT (1HO,4X,16HSEARCH COMPLETE.,I5,22H EVALUATIONS PERFORMED)      APP05430
      END                                                                   APP05440
C     SUBROUTINE ADS003 (ICALC,X1,Y1,YP1,X2,Y2,X3,Y3,X4,Y4,ZRO,AZ,A1,A2,   APP05450
C    * A3)                                                                  APP05460
C     ***************************************************************     APP05470
C     CUBIC POLYNOMIAL APPROXIMATION.                                       APP05480
C     ***************************************************************     APP05490
C                                                                           APP05500
C     BY G. N. VANDERPLAATS                              AUG. 1980          APP05510
C     NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                           APP05520
C                                                                           APP05530
C --- INPUT                                                                 APP05540
C     ICALC - CALCULATION CONTROL.  IF ICALC=1, DO THREE-POINT              APP05550
C             CURVE FIT.  IF ICALC=2, DO FOUR-POINT CURVE FIT.              APP05560
C     X1,Y1,YP1 - FIRST X-Y PAIR AND SLOPE.                                 APP05570
C                 Y1 AND YP1 ARE FUNCTION VALUE AND SLOPE.                  APP05580
C                 IF ICALC=2, YP1 IS NOT USED.                              APP05590
C     X2,Y2 - SECOND X-Y PAIR.                                              APP05600
C     X3,Y3 - THIRD X-Y PAIR.                                               APP05610
C     X4,Y4 - FOURTH X-Y PAIR.  NOT USED IF ICALC=1.                        APP05620
C     ZRO - SMALL POSITIVE NUMBER TO INDICATE NUMERICAL ZERO.               APP05630
C         INDICATOR OF MACHINE ACCURACY.                                    APP05640
C                                                                           APP05650
C --- OUTPUT                                                                APP05660
C     AZ,A1,A2,A3 - COEFICIENTS OF INTERPOLATING POLYNOMIAL.                APP05670
C                                                                           APP05680
C     APPROXIMATING FUNCTION IS                                             APP05690
C         Y = AZ + A1*X + A2*X**2 + A3*X**3                                 APP05700
C                                                                           APP05710
      SUBROUTINE ADS003 (ICALC,X1,Y1,YP1,X2,Y2,X3,Y3,X4,Y4,ZRO,AZ,A1,A2,   APP05720
     * A3)                                                                  APP05730
C                                                                           APP05740
C     EVALUTE COEFICIENTS.                                                  APP05750
C     CONSTANTS.                                                            APP05760
      X21=X2-X1                                                             APP05770
      X31=X3-X1                                                             APP05780
      X32=X3-X2                                                             APP05790
```

```
      X11=X1**2                                                          APP05800
      X22=X2**2                                                          APP05810
      X33=X3**2                                                          APP05820
      IF (ICALC.GT.1) GO TO 10                                           APP05830
C     THREE-POINT APPROXIMATION, GIVEN (X1,Y1,YP1), (X2,Y2), (X3,Y3).   APP05840
      A3=(X21*(Y3-Y1)/X31-X31*(Y2-Y1)/X21+X32*YP1)/(X21*X31*X32)         APP05850
      A2=((Y2-Y1)/X21-YP1)/X21-(2.*X1+X2)*A3                             APP05860
      A1=YP1-2.*A2*X1-3.*A3*X1*X1                                        APP05870
      AZ=Y1-X1*(A1+X1*(A2+A3*X1))                                        APP05880
      RETURN                                                             APP05890
C     FOUR-POINT APPROXIMATION, GIVEN (X1,Y1), (X2,Y2), (X3,Y3),        APP05900
C     (X4,Y4).                                                           APP05910
10    CONTINUE                                                           APP05920
      X42=X4-X2                                                          APP05930
      X41=X4-X1                                                          APP05940
      Q1=X33*X3*X21-X22*X2*X31+X11*X1*X32                                APP05950
      Q2=(X4**3)*X21-X22*X2*X41+X11*X1*X42                               APP05960
      Q3=X32*X21*X31                                                     APP05970
      Q4=X42*X21*X41                                                     APP05980
      Q5=Y3*X21-Y2*X31+Y1*X32                                            APP05990
      Q6=Y4*X21-Y2*X41+Y1*X42                                            APP06000
      DENOM=Q2*Q3-Q1*Q4                                                  APP06010
      A3=0.                                                              APP06020
      IF(ABS(DENOM).GT.ZRO) A3=(Q3*Q6-Q4*Q5)/DENOM                       APP06030
      A2=(Q5-Q1*A3)/Q3                                                   APP06040
      A1=(Y2-Y1)/X21-(X22*X2-X11*X1)*A3/X21-(X1+X2)*A2                   APP06050
      AZ=Y1-X1*(A1+X1*(A2+X1*A3))                                        APP06060
      RETURN                                                             APP06070
      END                                                                APP06080
C     SUBROUTINE ADS005(ICALC,A1,A2,A3,XMN1,XMX1)                        APP06090
CCCCC ***********************************************************       APP06100
CCCCC FIND MINIMUM(S) OF POLYNOMIAL TO 3RD ORDER.                        APP06110
CCCCC ***********************************************************       APP06120
CCCCC                                                                    APP06130
CCCCC BY G. N. VANDERPLAATS                              AUG. 1980       APP06140
CCCCC NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                        APP06150
CCCCC                                                                    APP06160
CCCCC --- INPUT                                                          APP06170
CCCCC ICALC - CALCULATION CONTROL.                                       APP06180
CCCCC         ICALC=1, FIRST ORDER POLYNOMIAL.  NO MINIMUM OR            APP06190
CCCCC         MAXIMUM EXISTS.  RETURN.                                   APP06200
CCCCC         ICALC=2, SECOND ORDER POLYNOMIAL.  FIND MINIMUM OR         APP06210
CCCCC                  MAXIMUM.                                          APP06220
CCCCC         ICALC=3, THIRD ORDER POLYNOMIAL.  FIND MINIMUM AND MAXIMUMAPP06230
CCCCC              IF THEY ARE REAL.                                     APP06240
CCCCC A1, A2, A3 - COEFICIENTS OF THE POLYNOMIAL,                        APP06250
CCCCC      Y = AZ + A1*X + A2*X**2 + A3*X**3.                            APP06260
C     NOTE - AZ IS NOT NEED IN CALCULATIONS, SO IS NOT BROUGHT           APP06270
```

```
C          THROUGH THE PARAMETER LIST.                                          APP06280
C                                                                               APP06290
C --- OUTPUT                                                                    APP06300
C     XMN1, XMX1 - VALUE OF X AT WHICH Y IS MINIMUM AND/OR MAXIMUM.             APP06310
C          IF ICALC = 1, XMN1 AND XMX1 ARE NOT FOUND.                           APP06320
C          IF ICALC = 2, XMN1 OR XMX1 IS FOUND, BUT NOT BOTH.                   APP06330
C          IF ICALC = 3, XMN1 AND XMX1 ARE FOUND.                               APP06340
C                                                                               APP06350
C --- NOTE.  IF XMN1 OR XMX1 IS NOT FOUND, OR IS IMAGINARY, A                   APP06360
C     VALUE OF -1.0E+20 IS RETURNED.                                            APP06370
C                                                                               APP06380
      SUBROUTINE ADS005 (ICALC,A1,A2,A3,XMN1,XMX1)                              APP06390
C                                                                               APP06400
C     INITIALIZE OUTPUT PARAMETERS.                                             APP06410
      XMN1=-1.0E+20                                                             APP06420
      XMX1=-1.0E+20                                                             APP06430
C     ICALC = 1.                                                                APP06440
C     LINEAR APPROXIMATION.  NO MINIMUM OR MAXIMUM EXISTS.  RETURN.             APP06450
      IF (ICALC.LT.2) RETURN                                                    APP06460
      IF (ICALC.GT.2) GO TO 20                                                  APP06470
C     ICALC = 2.                                                                APP06480
C     QUADRATIC APPROXIMATION.  FIND EITHER XMN1 OR XMX1.                       APP06490
10    XX=-.5*A1/A2                                                              APP06500
      IF (A2.GT.0.) XMN1=XX                                                     APP06510
      IF (A2.LT.0.) XMX1=XX                                                     APP06520
      RETURN                                                                    APP06530
20    CONTINUE                                                                  APP06540
C     ICALC = 3.                                                                APP06550
C     IF A3 IS NEAR ZERO, USE QUADRATIC INTERPOLATION.                          APP06560
      IF (ABS(A3).LT.1.0E-6) GO TO 10                                           APP06570
C     CUBIC APPROXIMATION.  FIND XMN1 AND XMX1.                                 APP06580
      B=A2**2-3.*A1*A3                                                          APP06590
      IF (B.LT.-1.0E-6) GO TO 30                                                APP06600
      IF (B.LT.0.) B=0.                                                         APP06610
      B=SQRT(B)                                                                 APP06620
      XMN1=(-A2+B)/(3.*A3)                                                      APP06630
      XMX1=(-A2-B)/(3.*A3)                                                      APP06640
30    CONTINUE                                                                  APP06650
      RETURN                                                                    APP06660
      END                                                                       APP06670
C     SUBROUTINE ADS006(X,F,N,XMULT,XADD,XNEW)                                  APP06680
C     DIMENSION X(N),F(N)                                                       APP06690
C     ******************************************************************        APP06700
C     ROUTINE TO FIND VALUES OF X WHICH WILL BRACKET THE MINIMUM OF F.          APP06710
C     ******************************************************************        APP06720
C                                                                               APP06730
C     BY G. N. VANDERPLAATS                          AUG. 1980                  APP06740
C     NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                               APP06750
```

```
C                                                                       APP06760
C --- INPUT                                                             APP06770
C     X(N) - ARRAY CONTAINING N VALUES OF X, N.GE.1.                    APP06780
C     X IS ASSUMED TO BE MONOTONICALLY INCREASING.                      APP06790
C         REQUIRED DIMENSION OF X IS N.                                 APP06800
C     F(N) - ARRAY CONTAINING N FUNCTION VALUES CORRESPONDING TO        APP06810
C         THE VALUES OF X.                                              APP06820
C         REQUIRED DIMENSION OF F IS N.                                 APP06830
C     XMULT, XADD - UPDATING CONSTANTS IF X MUST BE INCREASED TO        APP06840
C         FIND AN UPPER BOUND.                                          APP06850
C         UPDATE FORMULA IS XNEW = (1.+XMULT)*X(N) + XADD.              APP06860
C         XMULT AND XADD MUST BE NON-NEGATIVE AND AT LEAST ONE OF THEM  APP06870
C         MUST BE POSITIVE.                                             APP06880
C     N - NUMBER OF X-F PAIRS ALREADY CONTAINED IN X AND F.  N.GE.1.    APP06890
C                                                                       APP06900
C --- OUTPUT                                                            APP06910
C     XNEW - -1.0 IF F(N).GE.F(N-1) WHERE N.GE.2.  THIS INDICATES       APP06920
C         THAT X CONTAINS BOUNDS ON F.                                  APP06930
C         XNEW = PROPOSED NEW X-VALUE IF F(N).LT.F(N-1) FOR N.GE.2 OR   APP06940
C         IF N=1.                                                       APP06950
C     XNEW = (1.+XMULT)*X(N)+XADD.                                      APP06960
C                                                                       APP06970
      SUBROUTINE ADS006 (X,F,N,XMULT,XADD,XNEW)                         APP06980
      DIMENSION X(1),F(1)                                               APP06990
C                                                                       APP07000
      XNEW=(1.+XMULT)*X(N)+XADD                                         APP07010
      IF (N.LE.1) RETURN                                                APP07020
      NM1=N-1                                                           APP07030
      IF (F(N).LT.F(NM1)) RETURN                                        APP07040
C     MINIMUM IS BOUNDED.                                               APP07050
      XNEW=-1.                                                          APP07060
      RETURN                                                            APP07070
      END                                                               APP07080
C     SUBROUTINE ADS007(XL,X1,X2,XU)                                    APP07090
C     ******************************************************************APP07100
C     ROUTINE TO CALCULATE INITIAL INTERIOR POINTS, X1 AND X2 FOR       APP07110
C     FINDING THE MINIMUM OF A FUNCTION USING THE GOLDEN SECTION METHOD.APP07120
C     ******************************************************************APP07130
C                                                                       APP07140
C     BY G. N. VANDERPLAATS                              AUG. 1980      APP07150
C     NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                       APP07160
C                                                                       APP07170
C --- INPUT                                                             APP07180
C     XL - LOWER BOUND ON SOLUTION.                                     APP07190
C     XU - UPPER BOUND ON SOLUTION.                                     APP07200
C                                                                       APP07210
C --- OUTPUT                                                            APP07220
C     X1 - FIRST INTERMEDIATE POINT.  X1=GOLD*(XU-XL).                  APP07230
```

```
C      X2 - SECOND INTERMEDIATE POINT.  X2=XU-GOLD*(XU-XL).                    APP07240
CC           GOLD IS THE GOLDEN SECTION NUMBER (3-SQRT(5))/2.                  APP07250
C                                                                              APP07260
       SUBROUTINE ADS007 (XL,X1,X2,XU)                                         APP07270
       GOLDX=0.38196601*(XU-XL)                                                APP07280
       X1=XL+GOLDX                                                             APP07290
       X2=XU-GOLDX                                                             APP07300
       RETURN                                                                  APP07310
       END                                                                     APP07320
C      SUBROUTINE ADS008(XL,YL,X1,Y1,X2,Y2,XU,YU,IX)                           APP07330
CCCCC  ************************************************************************APP07340
CCCCC  ROUTINE TO UPDATE THE BOUNDS ON THE MINIMUM AND ESTIMATE A NEW          APP07350
CCCCC  INTERIOR POINT FOR THE GOLDEN SECTION METHOD.                           APP07360
CCCCC  ************************************************************************APP07370
CCCCC                                                                          APP07380
CCCCC  BY G. N. VANDERPLAATS                                 AUG. 1980         APP07390
CCCCC  NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                             APP07400
CCCCC                                                                          APP07410
CCCCC --- INPUT                                                                APP07420
CCCCC      XL,YL - LOWER BOUND AND FUNCTION VALUE.                             APP07430
CCCCC      X1,Y1 - FIRST INTERMEDIATE POINT AND FUNCTION VALUE.                APP07440
CCCCC      X2,Y2 - SECOND INTERMEDIATE POINT AND FUNCTION VALUE.               APP07450
CCCCC      XU,YU - UPPER BOUND AND FUNCTION VALUE.                             APP07460
CCCCC                                                                          APP07470
CCCCC --- OUTPUT                                                               APP07480
CCCCC      XL,YL - UPDATED XL AND YL.                                          APP07490
CCCCC      X1,Y1 - UPDATED X1 AND Y1 IF IX=2.  X1 ONLY IF IX=1.                APP07500
CCCCC      X2,Y2 - UPDATED X2 AND Y2 IF IX=1.  X2 ONLY IF IX=2.                APP07510
CCCCC      XU,YU - UPDATED XU AND YU.                                          APP07520
CCCCC      IX - NUMBER IDENTIFYING NEW INTERIOR POINT WHICH WAS CREATED.       APP07530
CCCCC           IX=1 OR 2, CORRESPONDING TO X1 OR X2.  THE CORRESPONDING       APP07540
CCCCC         FUNCTION VALUE, Y1 OR Y2 MUST BE CALCULATED LATER.               APP07550
C                                                                              APP07560
       SUBROUTINE ADS008 (XL,YL,X1,Y1,X2,Y2,XU,YU,IX)                          APP07570
C      GOLDEN SECTION FRACTION.                                                APP07580
       GOLD=0.38196601                                                         APP07590
C      PICK BEST POINT AND REDUCE BOUNDS.                                      APP07600
       IF (Y2.GT.Y1) GO TO 10                                                  APP07610
C      Y2.LT.Y1 .... X1 = NEW LOWER BOUND.                                     APP07620
       XL=X1                                                                   APP07630
       YL=Y1                                                                   APP07640
       X1=X2                                                                   APP07650
       Y1=Y2                                                                   APP07660
       X2=XU-GOLD*(XU-XL)                                                      APP07670
       IX=2                                                                    APP07680
       RETURN                                                                  APP07690
10     CONTINUE                                                                APP07700
C      Y1.LT.Y2 .... X2 = NEW UPPER BOUND.                                     APP07710
```

```
      XU=X2                                                              APP07720
      YU=Y2                                                              APP07730
      X2=X1                                                              APP07740
      Y2=Y1                                                              APP07750
      X1=XL+GOLD*(XU-XL)                                                 APP07760
      IX=1                                                               APP07770
      RETURN                                                             APP07780
      END                                                                APP07790
C     SUBROUTINE ADS010(NDV,ALPHA,X,S,XNEW)                              APP07800
CC    ***************************************************************APP07810
CC    ROUTINE TO UPDATE THE X-VECTOR TO GIVE XNEW=X+ALPHA*S.             APP07820
CC    ***************************************************************APP07830
CC                                                                       APP07840
CC    BY G. N. VANDERPLAATS                          SEPT. 1980          APP07850
CC    NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIF.                        APP07860
CC                                                                       APP07870
CC --- INPUT                                                            APP07880
CC    NDV - NUMBER OF COMPONENTS OF X, S AND XNEW.                       APP07890
CC    ALPHA - SCALAR MULTIPLIER ON S-VECTOR.                             APP07900
CC    X - VECTOR OF DESIGN VARIABLES.                                    APP07910
CC        MINIMUM DIMENSION OF X IS NDV.                                 APP07920
CC    S - SEARCH VECTOR.                                                 APP07930
CC        MINIMUM DIMENSION OF S IS NDV.                                 APP07940
CC                                                                       APP07950
CC --- OUTPUT                                                           APP07960
CC    XNEW - NEX X-VECTOR.                                               APP07970
CC        MINIMUM DIMENSION OF XNEW IS NDV.                              APP07980
CC                                                                       APP07990
CC    XNEW(I) = X(I) + ALPHA*S(I), I=1,NDV.                              APP08000
C                                                                        APP08010
      SUBROUTINE ADS010 (NDV,ALPHA,X,S,XNEW)                             APP08020
      DIMENSION X(1),S(1),XNEW(1)                                        APP08030
C                                                                        APP08040
      DO 10 I=1,NDV                                                      APP08050
10    XNEW(I)=X(I)+ALPHA*S(I)                                            APP08060
      RETURN                                                             APP08070
      END                                                                APP08080
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Professor G. N. Vanderplaats, Code 69Vn Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	Professor D. Salinas, Code 69Sl Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	Lt. L. E. Madsen 457 Foam Street Monterey, California 93940	1

Thesis
M274865 Madsen
c.1
193682
Engineering design optimization by the augmented Lagrange multiplier method.